universität wien

MASTERARBEIT

Titel der Masterarbeit

„On the Word Problem for Groups“

Verfasser

Alexander Bors, BSc

angestrebter akademischer Grad

Master of Science (MSc)

Wien, 2014

Studienkennzahl lt. Studienblatt:	A 066 821
Studienrichtung lt. Studienblatt:	Mathematik
Betreuer:	Ao. Univ.-Prof. Dr. Karl Auinger

„The words. Why did they have to exist? Without them, there wouldn't be any of this.“
Markus Zusak, *The Book Thief*

Preface

In 1912, Max Dehn proposed three "fundamental problems for infinite discontinuous groups" in (5), exhibiting their importance not just for the theory of abstract infinite groups, but also for geometry. Among these three was what he called the "Identitätsproblem", now known as the "word problem": Given a group by means of a finite collection of generators and finitely many defining relations between them (i.e., relations such that any relation holding between them is a consequence of these), decide algorithmically in a finite number of computation steps whether an element of the group, given in terms of the specified generators, represents the identity element of the group or not.
This was still before the introduction of a formal concept of "algorithm" such as Turing machines by Alan Turing in (17) in 1936, meaning that there were no formal foundations for showing the nonexistence of algorithms for certain tasks. In 1955 then, Pyotr Novikov succeeded in finding an instance of the word problem as described above for which such an algorithm cannot exist; see (13). Boone found a different proof just three years later, see (3). The main focus of this thesis lies on this result, in modern language the existence of a "finitely presented group with undecidable word problem". However, we will not follow Novikov's or Boone's proof, but instead use a proof found by Valiev, published in (18) and also described in Lyndon's and Schupp's book (10), which can be seen as a fundament for big parts of this thesis.
Still, there is a lot of preliminary work to do. Since we assume some basic knowledge in group theory, but not in recursion theory, we will develop it by means of Turing machines from the scratch in the first chapter, "Recursion-Theoretic Foundations". The "highlight" of this chapter is the undecidability of the halting problem.
Valiev's proof also makes use of an astonishing and deep theorem by Matiyasevich relating number theory and recursion theory: Any recursively enumerable set of integers is Diophantine! This will be the topic of the second chapter, "Number-Theoretic Foundations". We also develop the theory there from the scratch, but instead of giving Matiyasevich's original proof of Diophantineness of the sequence of Fibonacci numbers with even index, we follow the one from his book (11), which is already a refined version of it, avoiding use of the preliminary work by Julia Robinson. The "highlight" of that chapter, apart from Matiyasevich's theorem, of course, is the undecidability of Hilbert's Tenth Problem.
We also need quite a few results and methods from so-called combinatorial group theory, the study of groups by means of generators and relations, presented in the third chapter, "Group-Theoretic Foundations". We formalize the idea of treating groups this way by means of the concept of a "group presentation" and consider groups constructed from given ones by certain manipulations on their presentations (more precisely, HNN extensions and free products with and without amalgamation). As a "highlight" of this chapter, we shall finally be able to formally define the notion of "word problem" as well as "generalized word problem".
The fourth chapter, "Undecidability of the Word Problem", is devoted not only to the construction of finite group presentations with undecidable word problem, but also to yet another deep theorem, this time relating recursion theory and group theory, the so-called Higman Embedding Theorem. The proof of it, building up on Valiev's Principal Lemma, can be seen as the "highlight" of this chapter.
Finally, in the fifth chapter, "Decidability of the Word Problem", we will be dealing with a selection of two classes of groups which are known to have decidable word problem - one-relator groups and hyperbolic groups. Whereas showing this for the first type of groups just

continues the combinatorial methods from the third chapter, the definition of the second type of groups already is essentially geometric, leading us into geometric group theory. The "highlight" of this final chapter are the stochastic results due to Gromov stating, intuitively, that "almost all" finite group presentations have decidable word problem.

Acknowledgements

This thesis would not have come into being without the course on group theory given by my supervisor Karl Auinger at Vienna University in the spring of 2013, which put a great focus on combinatorial and graph-theoretic methods in group theory and therefore raised my attention to this subject. It also provided a great basis for it, and I would like to thank Prof. Auinger for giving me much freedom in writing this thesis as well as his helpful comments throughout the work.
As for the curriculum vitae, I thank Thomas Quaritsch from TU Graz for his great model available under `http://latex.tugraz.at/vorlagen/diverse` which I modified for my own CV.
Many thanks also go to the user "Hennich" of the Internet forum "Matroids Matheplanet" (`http://www.matheplanet.at/`), who provided me with the idea of passing from Turing machines to register machines to convince anyone in doubt of it that Turing machines basically are capable of anything a computer can do, to be found at the end of the first chapter, in the subchapter on "The Church-Turing Thesis".
Also, I want to thank my parents for their constant support and understanding, not only, but in particular during the time in which this thesis was written.

A. Bors, June 2014

Contents

1 Recursion-Theoretic Foundations

In this chapter, we develop the recursion-theoretic concepts and results which are needed later. Readers familiar with recursion theory probably only need to skim through in order to learn the notation used in this thesis.

1.1 Formal Foundations

We denote by $\mathbb{N}$ (or alternatively, depending on the context, ω or $\aleph_0$) the set of all natural numbers, i.e., the smallest limit ordinal (in particular, $0 \in \mathbb{N}$). In order to define the notions of "relation" and "function" formally, we use Kuratowski's definition of an ordered pair, i.e.: $\langle x, y\rangle := \{\{x\}, \{x, y\}\}$. Then, for a function f, let $\mathrm{dom}(f) := \{x \mid \exists y : \langle x, y\rangle \in f\}$ denote the **domain** and $\mathrm{ran}(f) := \{y \mid \exists x : \langle x, y\rangle \in f\}$ the **range** of f. For sets A, B the so-called **set power** B^A denotes the set of all functions $f : A \to B$. In particular, for an arbitrary set A and a natural number $k \in \omega$, the set power A^k denotes the set of all **k-tuples** $(a_0, \ldots, a_{k-1})$ over A, that is, functions with domain $k = \{0, \ldots, k-1\}$ and range a subset of A. Note that also pairs written with round brackets are tuples in this sense, as opposed to the Kuratowski pairs $\langle x, y\rangle$.

The reason why we rather want to view tuples as functions with domain a natural number is because this makes it formally easy to view each set as an "alphabet": For any set X, let $X^* := \bigcup_{n\in\omega} X^n$ denote the so-called set of **words** or **strings over** X (and note that $X^0 = \{\varnothing\}$ by definition; $\varnothing$ is also called the **empty word** (over any X)). We then identify each $x \in X$ with the corresponding 1-tuple $\{\langle 0, x\rangle\} \in X^*$. It is easy to see that words over X possess "unique readability", i.e.: For any $w \in X^*$, there exists a unique $n \in \omega$ (namely the domain of w) and unique $x_0, \ldots, x_{n-1} \in X$ such that $w = (x_0, \ldots, x_{n-1})$. Furthermore, we define **concatenation** of words over X as a binary operation $\cdot : (X^*)^2 \to X^*$, $(w_1, w_2) \mapsto \cdot(w_1, w_2) := w_1 w_2$, where $\mathrm{dom}(w_1 w_2) = \mathrm{dom}(w_1) + \mathrm{dom}(w_2)$ (addition of ordinals or natural numbers), and for $k \in \mathrm{dom}(w_1) + \mathrm{dom}(w_2)$: $(w_1 w_2)(k) := \begin{cases} w_1(k), & \text{if } k \in \mathrm{dom}(w_1) \\ w_2(k - \mathrm{dom}(w_1)) & \text{else} \end{cases}$. By the identification of elements of X with 1-tuples described above, we can then write a word $(x_0, \ldots, x_{n-1})$ over X also as $x_0 \cdots x_{n-1}$. The structure $(X^*, \cdot, \varnothing)$ is, as is readily checked, a monoid, but we will not deal with the properties of this monoid before the third chapter.

Moreover, a **directed graph** (or **digraph**) is, at least in this chapter, understood to be a pair $\Gamma = (V, E)$, where $E \subseteq V^2$. The elements of V are called **vertices**, those of E **edges of** Γ. For each $e \in E$, we denote by $\iota(e)$ the first entry of the pair e, called **initial vertex of** e, and by $\tau(e)$ the second entry, called **terminal vertex of** e. A digraph (V, E) is called **finite** if and only if V is finite. A **(directed) path in** Γ is a sequence $p \in E^* \cup E^\omega$ such that for all $n \in \omega$ with $n + 1 \in \mathrm{dom}(p)$: $\tau(p(n)) = \iota(p(n+1))$. For $p \neq \varnothing$, we further denote by $\iota(p) := \iota(p(0))$ the **initial vertex of** p, and, if also $\mathrm{dom}(p) < \omega$, by $\tau(p) := \tau(p(\mathrm{dom}(p) - 1))$ the **terminal vertex of** p. Furthermore, by definition, every vertex of Γ is considered an initial and a terminal vertex of the "empty path" $\varnothing$.

For digraphs $\Gamma_1 = (V_1, E_1), \Gamma_2 = (V_2, E_2)$, an **isomorphism between Γ_1 and Γ_2** is a bijection $\varphi : V_1 \to V_2$ such that for all $v, w \in V_1$: $(v, w) \in E_1 \Leftrightarrow (\varphi(v), \varphi(w)) \in E_2$. Of course, Γ_1 and Γ_2 are called **isomorphic** if and only if there exists an isomorphism between them.

For composition of Turing machines, we will also consider **rooted labeled digraphs**. These are tuples $(V, E, \lambda_V, \lambda_E, v)$ such that (V, E) is a digraph, λ_X is a function with domain X for $X \in \{V, E\}$ and $v \in V$. When $\mathrm{ran}(\lambda_X) \subseteq \Lambda_X$ for $X \in \{V, E\}$, we also call the digraph (Λ_V, Λ_E)-

labeled. v is called the **distinguished vertex**. A rooted labeled digraph $(V, E, \lambda_V, \lambda_E, v)$ is called **finite** if and only if the underlying digraph (V, E) is finite. A rooted labeled digraph is called **folded** if and only if for all edges $e_1, e_2 \in E$ with $\iota(e_1) = \iota(e_2)$, we have the implication $\lambda_E(e_1) = \lambda_E(e_2) \Rightarrow e_1 = e_2$ (i.e., there are no different edges sharing a common intial vertex and carrying the same label).
Finally, we shall use the following notations: For a function f and a set M, we denote by $f \restriction M := \{\langle x, y\rangle \in f \mid x \in M\}$ the **restriction of f to M**. If $\vec{f} = (f_0, \ldots, f_{k-1})$ is a k-tuple of functions with common domain D, and $\vec{x} = (x_0, \ldots, x_{k-1}) \in D^k$, we let $\vec{f}(\vec{x}) := (f_0(x_0), \ldots, f_{k-1}(x_{k-1}))$. A function f from a subset of some set X to a set Y is called a **partial function from X to Y**, and we write $f : X \dashrightarrow Y$.

1.2 Basic Turing Machine Recursion Theory

This section is based on (8) and (12). First, we shall describe the heuristic idea behind the concept of a "Turing machine" in order to motivate the formal definition given afterward.
A (heuristic) Turing machine is a device operating on finitely many so-called "work tapes". We imagine these as linear and in both directions infinite (of course, this is an idealization of the practical situation) arrays of so-called "cells", which are the smallest parts of work tapes carrying a full piece of information. The information on a work tape comes from printing symbols from a given finite alphabet on it (one symbol per cell); for convenience, we assume that there are no empty (i.e, not imprinted) cells on a work tape, but instead, we distinguish one symbol $*$ of X as the "empty" (or "improper") symbol (in particular, we assume that all alphabets used are nonempty) and set $X_0 := X \backslash \{*\}$.
The work tapes may (and usually will) be imprinted before applying the Turing machine, but always such that almost all cells are imprinted with $*$. The machine has so-called "heads" (one per work tape) that are placed over certain cells at the beginning and, at each step, read the imprinting of the cell over which they currently are.
Furthermore, a Turing machine can take different "states" in the course of its operations on the work tapes. What exactly the machine does at each step is not only determined by what the heads read, but also by the state the machine is in (however, these are all the influential parameters). The "active" part of each work step then consists firstly in a possible change of the imprinting of the cells at the current head positions, secondly in moving the heads (each head can either remain where it is or be moved one cell to the left or to the right), and thirdly in changing the machine state.
Of course, the machine must already be in some state after "booting up"; since it should be a completely deterministic device, this state must always be the same, and it is called the "start state" of the machine. Moreover, there is always another distinguished state, distinct from the start state, which is called the "halting state". Once the machine reaches this state, it immediately stops.
After all these heuristic observations, let us now give the formal definition. For this, we identify a (heuristic) Turing machine with its so-called "rules table" (which one could compare with the source code of modern computer programs), which consists of all clauses what to do in each possible situation.

Definition 1.2.1. *Let X be a finite alphabet, $m = |X|$, $S \subseteq \mathbb{N}$ finite, with $l = |S| \geqslant 2$ and $k \in \mathbb{N}$,$k \geqslant 1$. A **Turing machine over X with k work tapes and set of states S** (shortly: **(X, k, S)-TM**) is an $((l-1)m^k \times 5)$-matrix $\mathbb{A}$ of the form*

$$\begin{pmatrix}
s_1 & \overrightarrow{x_1} & \overrightarrow{y_1}^{(1)} & \overrightarrow{t_1}^{(1)} & s_1^{(1)} \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
s_1 & \overrightarrow{x_{m^k}} & \overrightarrow{y_{m^k}}^{(1)} & \overrightarrow{t_{m^k}}^{(1)} & s_{m^k}^{(1)} \\
s_2 & \overrightarrow{x_1} & \overrightarrow{y_1}^{(2)} & \overrightarrow{t_1}^{(2)} & s_1^{(2)} \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
s_2 & \overrightarrow{x_{m^k}} & \overrightarrow{y_{m^k}}^{(2)} & \overrightarrow{t_{m^k}}^{(2)} & s_{m^k}^{(2)} \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
s_{l-1} & \overrightarrow{x_1} & \overrightarrow{y_1}^{(l-1)} & \overrightarrow{t_1}^{(l-1)} & s_1^{(l-1)} \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
s_{l-1} & \overrightarrow{x_{m^k}} & \overrightarrow{y_{m^k}}^{(l-1)} & \overrightarrow{t_{m^k}}^{(l-1)} & s_{m^k}^{(l-1)}
\end{pmatrix},$$

where $S = \{s_1, \ldots, s_l\}, X^k = \{\overrightarrow{x_1}, \ldots, \overrightarrow{x_{m^k}}\}$ *and* $\overrightarrow{y_j}^{(i)} \in X^k, \overrightarrow{t_j}^{(i)} \in \{-1, 0, 1\}, s_j^{(i)} \in S$ *for* $i = 1, \ldots, l-1$ *and* $j = 1, \ldots, m^k$. *The elements of* S *are called* ***states of*** $\mathbb{A}$, s_1 *is called the* ***start state***, s_l *the* ***halting state of*** $\mathbb{A}$.

Remark 1.2.2. As indicated earlier, the rows of a (formal) Turing machine are to be understood as clauses. The row $\begin{pmatrix} s_i & \overrightarrow{x_j} & \overrightarrow{y_j}^{(i)} & \overrightarrow{t_j}^{(i)} & s_j^{(i)} \end{pmatrix}$ is to be read as: In state s_i, and when the heads read the tuple $\overrightarrow{x_j}$, change the imprinting of the head cells to the tuple $\overrightarrow{y_j}^{(i)}$, then move the heads according to the tuple $\overrightarrow{t_j}^{(i)}$ (0 meaning "no movement", $+1$ "one cell to the right" and -1 "one cell to the left"), and finally, change the state to $s_j^{(i)}$. The start state is distinguished by the fact that it is the first state mentioned in the matrix and the halting state by its nonappearance in the first column of the matrix. Of course, we also do not need to specifiy any clauses for what to do in the halting state as the machine immediately stops when reaching it anyway.

Example 1.2.3. Let $X = \{x_1, \ldots, x_m\}$ with $|X| = m$ be a finite alphabet, $k \in \mathbb{N}$, $k \geqslant 1$ and $j \in \{1, \ldots, k\}$. We now define three very simple, but important types of Turing machines, the so-called **elementary Turing machines**:

(1) $r_X^{(j)}$, the so-called **elementary right machine over** X **on tape** j; it has states 0,1 and 2, where 0 is the start and 2 the halting state. The first m^k rows of $r_X^{(j)}$ are of the form $\begin{pmatrix} 0 & \vec{w} & \vec{w} & \begin{pmatrix} 0 \\ \vdots \\ 0 \\ +1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} & 1 \end{pmatrix}$, where $\vec{w}$ runs over all the elements of X^k and the $+1$ in the vector in the second to last column is its j-th entry. The last m^k rows are of the form $\begin{pmatrix} 1 & \vec{w} & \vec{w} & \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} & 2 \end{pmatrix}$, where again, $\vec{w}$ runs over the words in X^k.

(2) $l_X^{(j)}$, the **elementary left machine over X on tape j** is defined analogously, replacing $+1$ by -1 in the vector in the second-to-last column in the first m^k rows.
(3) Let $x \in X$. The **elementary printing machine over X with respect to x on tape j**, in symbols $p_X^{(j,x)}$, has states 0 and 1, with 0 the start and 1 the halting state. Its m^k rows are of the form $\begin{pmatrix} 0 & \begin{pmatrix} y_1 \\ \vdots \\ y_k \end{pmatrix} & \begin{pmatrix} y_1 \\ \vdots \\ y_{j-1} \\ x \\ y_{j+1} \\ \vdots \\ y_k \end{pmatrix} & \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} & 1 \end{pmatrix}$, where $y_1, \ldots, y_k \in X$.
Two remarks about that:
Firstly, all three types of Turing machines are, strictly speaking, not well-defined this way, but only after choosing a linear order on X^k. However, it is clear that all the different matrices one can get this way are in some sense "equivalent" - see the next Definition 1.2.4.
Secondly, it is, from a heuristic point of view, easy to see what these machines "do": $r_X^{(j)}$ (resp. $l_X^{(j)}$), from any start configuration, just performs a single move to the right (resp. left) on the j-th work tape without moving the other heads or changing the imprinting on any of the work tapes. After that, it has one additional work step, where it does nothing at all and then stops; we defined it this way, not letting it stop right after the first work step, because this version is more useful in compositions (see Definition 1.2.8). The elementary printing machine $p_X^{(j,x)}$ just prints the symbol x on the j-th tape start configuration cell, changes nothing on the other tapes, does not move its heads and immediately halts.
However, so far, these are all only heuristic observations, since on the formal level, we have only defined what a Turing machine "is", and not what it "does". Let us do this right now.

Definition 1.2.4. *Let X be a finite alphabet, $S \subseteq \mathbb{N}$ finite with $|S| \geqslant 2$ and $k \in \mathbb{N}$ with $k \geqslant 1$.*
*(1) An (X,k,S)-**configuration** is a triple $(s, \vec{j}, \vec{\alpha})$, where $s \in S, \vec{j} \in \mathbb{Z}^k, \vec{\alpha} \in (X^{\mathbb{Z}})^k$. We denote the set of all (X,k,S)-configurations by $\mathcal{C}_{(X,k,S)}$.*
For the rest of this definition, let $\mathbb{A}$ be an (X,k,S)-TM.
*(2) Let $(s_1, \overrightarrow{j_1}, \overrightarrow{\alpha_1}), (s_2, \overrightarrow{j_2}, \overrightarrow{\alpha_2})$ be (X,k,S)-configurations. We call $(s_2, \overrightarrow{j_2}, \overrightarrow{\alpha_2})$ the **successor configuration of** $(s_1, \overrightarrow{j_1}, \overrightarrow{\alpha_1})$ **under** $\mathbb{A}$, in symbols $(s_1, \overrightarrow{j_1}, \overrightarrow{\alpha_1})\Sigma_{\mathbb{A}}(s_2, \overrightarrow{j_2}, \overrightarrow{\alpha_2})$, if and only if $\mathbb{A}$ contains the row $\begin{pmatrix} s_1 & \overrightarrow{\alpha_1}(\overrightarrow{j_1}) & \vec{y} & \vec{t} & s_2 \end{pmatrix}$, where $\overrightarrow{\alpha_1} \restriction (\mathbb{Z}^k \backslash \{\overrightarrow{j_1}\}) = \overrightarrow{\alpha_2} \restriction (\mathbb{Z}^k \backslash \{\overrightarrow{j_1}\})$ and $\overrightarrow{\alpha_2}(\overrightarrow{j_1}) = \vec{y}$, as well as $\overrightarrow{j_2} = \overrightarrow{j_1} + \vec{t}$.*
*(3) The **configuration graph of $\mathbb{A}$ (with respect to S)** is the digraph $(\mathcal{C}_{(X,k,S)}, \Sigma_{\mathbb{A}})$. If also $S' \subseteq \mathbb{N}$ finite with $|S'| \geqslant 2$, then $\mathbb{A}$ and an (X,k,S')-TM $\mathbb{A}'$ are called **equivalent** if and only if there exists a bijection $f : S \to S'$ mapping the start (resp. halting) state of $\mathbb{A}$ to the start (resp. halting) state of $\mathbb{A}'$ such that the map $\varphi_f : \mathcal{C}_{(X,k,S)} \to \mathcal{C}_{(X,k,S')}, (s, \vec{j}, \vec{\alpha}) \mapsto (f(s), \vec{j}, \vec{\alpha})$ is an isomorphism between the configuration graphs of $\mathbb{A}$ and $\mathbb{A}'$ (with respect to S and S' respectively). A **partial run of** $\mathbb{A}$ is a nonempty directed path in the configuration graph of $\mathbb{A}$, and a **(complete) run of** $\mathbb{A}$ is a partial run of $\mathbb{A}$ whose initial vertex has as first entry the start state of $\mathbb{A}$, and which is either infinite or, if it is finite, whose terminal vertex has as first entry the halting state of $\mathbb{A}$.*
For the last two parts of the definition, let $w = x_1 \cdots x_n \in X_0^*$.
*(4) The **start configuration of $\mathbb{A}$ on** w is the following (X,k,S)-configuration $(s, \vec{j}, \vec{\alpha})$: s*

is the start state of $\mathbb{A}$, $\vec{j} = (0, \ldots, 0)$, $\alpha_0(i) = \begin{cases} x_i, & \text{if } i \in \{1, \ldots, n\} \\ * & \text{else} \end{cases}$ *and* $\alpha_r(i) = *$ *for all* $r \in \{1, \ldots, k-1\}$ *and all* $i \in \mathbb{Z}$. *The* ***(complete) run of*** $\mathbb{A}$ ***on*** w *is then defined as the (uniquely determined) run of* $\mathbb{A}$ *whose initial vertex is the start configuration of* $\mathbb{A}$ *on* w.
(5) If the run of $\mathbb{A}$ *on* w *is a finite path, we say that* $\mathbb{A}$ ***halts on (input)*** w, *and in this case, the* ***output of*** $\mathbb{A}$ ***on*** w *is defined as* $(\alpha_k(1), \ldots, \alpha_k(m_0))$, *where* $m_0 = \max\{m \in \mathbb{N} \mid \forall i \in \{1, \ldots, m\} : \alpha_k(i) \neq *\}$.

Remark 1.2.5. (1) Configurations are to be understood as "snapshots" during the runs of Turing machines; $(s, \vec{j}, \vec{\alpha})$ represents the situation where the Turing machine is in state s, the heads are on the positions given by the tuple $\vec{j}$ and the work tapes have the imprintings given by the tuple of functions (assignments) $\vec{\alpha}$.
(2) With notation as in point (3) of Definition 1.2.4, it is easy to see that $\mathbb{A}$ and $\mathbb{A}'$ are equivalent if and only if there exists a bijection $f : S \to S'$ such that the matrix $\mathbb{A}'$ arises from $\mathbb{A}$ by replacing each row $\begin{pmatrix} s & \vec{x} & \vec{y} & \vec{t} & s' \end{pmatrix}$ by $\begin{pmatrix} f(s) & \vec{x} & \vec{y} & \vec{t} & f(s') \end{pmatrix}$ (note that the configuration graph of $\mathbb{A}$ has an edge between $(s_0, \overrightarrow{j_0}, \overrightarrow{\alpha_0})$ and $(s_2, \overrightarrow{j_1}, \overrightarrow{\alpha_1})$ if and only if $\mathbb{A}$ contains the row $\begin{pmatrix} s_0 & \overrightarrow{\alpha_0}(\overrightarrow{j_0}) & \overrightarrow{\alpha_1}(\overrightarrow{j_0}) & \overrightarrow{j_1} - \overrightarrow{j_0} & s_1 \end{pmatrix}$).
(3) Point (4) of Definition 1.2.4 fixes the standard start configuration for working with different inputs. See also the next Definition 1.2.6.
(4) m_0 in point (5) of Definition 1.2.4 is well-defined, since the set over which the maximum is taken is on the one hand nonempty (as 0 is an element) and on the other hand bounded by the path length of the run of $\mathbb{A}$ on w.

Definition 1.2.6. *Let* X *be a finite alphabet such that* $0, 1 \in X_0$. *A partial function* $f : X_0^* \dashrightarrow X_0^*$ *is called* ***(Turing-)computable*** *if and only if there exists a finite subset* $S \subseteq \mathbb{N}$ *with* $|S| \geqslant 2$, *a number* $k \in \mathbb{N}$ *with* $k \geqslant 1$ *and an* (X, k, S)*-TM* $\mathbb{A}$ *which* ***computes*** f, *i.e.:* $\mathbb{A}$ *halts on all inputs, and on inputs* $w \in X_0^* \backslash \mathrm{dom}(f)$, *the output of* $\mathbb{A}$ *is* 0, *and else, the output is the concatenation* $1 \cdot f(w)$.

Example 1.2.7. Let X be the finite alphabet $\{*, 0, 1\}$. We show that the "turn around" function $f : X_0^* \to X_0^*, y_1 \cdots y_n \mapsto y_n \cdots y_1$, is computable. By definition, we have to specify a finite subset $S \subseteq \mathbb{N}$ with $|S| \geqslant 2$, a number $k \in \mathbb{N}$ with $k \geqslant 1$ and an (X, k, S)-TM $\mathbb{A}$ which halts on every input $y_1 \cdots y_n \in X_0^*$, with output $1 y_n \cdots y_1$. We set $k := 2$ and $S := \{0, 1, 2, 3\}$; for better readability, we set $s_0 := 0, s_r := 1, s_l := 2, s_1 := 3$.
Now we need to write down a suitable matrix $\mathbb{A}$. In order to simplify notation, we make the following convention: When defining Turing machines which shall compute some partial function, we are allowed to omit rows which stand for clauses concerning situations which do not occur in any of the runs of the machine on all possible inputs. Also, we want to have more freedom when it comes to the ordering of the rows, writing them down in a different order than the standard one when this is more intuitive. In this example, even under these omissions, all states different from the halting state are mentioned in the first column, but one should be aware that this need not happen, in which case one needs to specify the halting state separately.
Coming back to our example: It is easy to see that the Turing machine $\mathbb{A}$ specified below does what we want: First, it moves both heads one step to the right. If the input is empty, the second head just prints 1 and the machine halts immediately. Otherwise, the first head goes to the right end of the input, moving as long as it reads a proper symbol, and the second head keeps printing 1 on cell number 1 of the second tape. When the first head has reached

the end of the input, it starts moving to the left again, "telling" the second head the input symbols in inverse order which are then printed one after another on the second tape. After this, the machine halts.

$$\mathbb{A} := \left(\begin{array}{cccc}
s_0 & \binom{*}{*} & \binom{*}{*} & \binom{+1}{+1} & s_r \\
s_r & \binom{*}{*} & \binom{*}{1} & \binom{0}{0} & s_1 \\
s_r & \binom{0}{*} & \binom{0}{1} & \binom{+1}{0} & s_r \\
s_r & \binom{1}{*} & \binom{1}{1} & \binom{+1}{0} & s_r \\
s_r & \binom{0}{1} & \binom{0}{1} & \binom{+1}{0} & s_r \\
s_r & \binom{1}{1} & \binom{1}{1} & \binom{+1}{0} & s_r \\
s_r & \binom{*}{1} & \binom{*}{1} & \binom{-1}{+1} & s_l \\
s_l & \binom{0}{*} & \binom{0}{0} & \binom{-1}{+1} & s_l \\
s_l & \binom{1}{*} & \binom{1}{1} & \binom{-1}{+1} & s_l \\
s_l & \binom{*}{*} & \binom{*}{*} & \binom{0}{0} & s_1
\end{array} \right).$$

Next, we will talk about composition of Turing machines, which allows us to "build" a larger Turing machine out of several small ones, all of which fulfill different purposes in the construction. As an example, consider the following question: What if, when defining the notion of "computable partial function" (see Definition 1.2.6), we had not worked with a standard start configuration, but instead, demanded that there exist, for S, k suitable, an (X, k, S)-TM $\mathbb{A}$ such that whenever we print some input $w \in X_0^* \backslash \{\varnothing\}$ somewhere on the first work tape of $\mathbb{A}$, and we place the first head of $\mathbb{A}$ anywhere on the first work tape (and the others at their "usual positions", i.e., cell number 0 on the corresponding tape) before starting the run of $\mathbb{A}$, it will always halt with output either 0 (in case $w \notin \mathrm{dom}(f)$) or $1 \cdot f(w)$? Let us call partial functions f for which this holds **strongly (Turing-)computable** (note that this definition is not standard).
Of course, formally the notion of "strongly computable" is stronger than "computable", but it will turn out that it really is not. The idea how to see this is to build, out of a Turing machine $\mathbb{A}$ computing some partial function f, a bigger Turing machine which first runs a "search routine" to find the nonempty input on the first work tape, get into the standard start configuration and then just run $\mathbb{A}$. This already is an easy example of composition of Turing machines, for which we now give the official definition, following the intuitive idea of a "flow-chart" for Turing machines:

Definition 1.2.8. *Let X be a finite alphabet, $k \in \mathbb{N}$, $k \geqslant 1$, and let $\mathcal{A}$ be a finite set such that for each $\mathbb{A} \in \mathcal{A}$, there exists a finite $S \subseteq \mathbb{N}$ with $|S| \geqslant 2$ such that $\mathbb{A}$ is an (X, k, S)-TM.*
*(1) A **flow-chart over** $\mathcal{A}$ is a finite folded rooted $(\mathcal{A}, X^k)$-labeled digraph.*

(2) Let $\mathcal{F} = (V, E, \lambda_V, \lambda_E, v)$ be a flow-chart over $\mathcal{A}$. Fix any linear order $<$ on V such that v is the smallest vertex. Then the ***Turing machine with respect to $\mathcal{F}$ (and $<$)****, in symbols $\mathcal{T}_{\mathcal{F}}$, is defined as follows:*

$$\mathcal{T}_{\mathcal{F}} = \begin{pmatrix} \mathbb{A}_1 \\ \mathbb{A}_2 \\ \vdots \\ \mathbb{A}_n \end{pmatrix},$$

where $n = |V|$, and $\mathbb{A}_i$ is a Turing machine obtained from the i-th (with respect to $<$) vertex $v_i \in V$ as follows: For $j = 1, \ldots, n$, let σ_j be the cardinality of the set of states of $\lambda_V(v_j)$, and let $\mathbb{A}_i' := \lambda_V(v_i)$. In order to obtain $\mathbb{A}_i$ from $\mathbb{A}_i'$, first replace each occurrence of the k-th state of $\mathbb{A}_i'$ (where we take the states of $\mathbb{A}_i'$ to be ordered as they appear in the first column, with the halting state the largest/last state) by $\sigma_1 + \ldots + \sigma_{i-1} + k$, for $k = 1, \ldots, \sigma_i$, giving the matrix $\mathbb{A}_i''$. Then, in $\mathbb{A}_i''$, replace in each row $\begin{pmatrix} s & \vec{x} & \vec{y} & \vec{t} & \sigma_1 + \ldots + \sigma_i \end{pmatrix}$ ending in the "new halting state" the last entry either by $\sigma_1 + \cdots + \sigma_{r-1} + 1$ if one of the edges of $\mathcal{F}$ with initial vertex v_i carries the label $\vec{y}$, where $r \in \{1, \ldots, n\}$ is such that v_r is the terminal vertex of the uniquely determined edge with initial vertex v_i and label $\vec{y}$, or else replace it by $\sigma_1 + \cdots + \sigma_n$. The matrix obtained after all these transformations is $\mathbb{A}_i$.

Remark 1.2.9. (1) With notation as in Definition 1.2.8, we now explain how $\mathcal{T}_{\mathcal{F}}$, viewed as an $(X, k, \{1, \ldots, \sigma_n\} \backslash \{\sigma_1, \sigma_1 + \sigma_2, \ldots, \sigma_1 + \cdots + \sigma_{n-1}\})$-TM, works: Its complete runs always start like a complete run of the machine $\lambda_V(v)$, the label of the distinguished vertex. However, whenever $\lambda_V(v)$ would halt, it is checked whether any of the edges of $\mathcal{F}$ with intial vertex v are labeled with the tuple printed in the last work step before halting. If so, then $\mathcal{T}_{\mathcal{F}}$ does not halt, but instead switches to the initial state of the machine labeling the terminal vertex of this uniquely determined edge, and goes on, repeating the same checking procedure for the new vertex before halting, and so on. If the result of the check is negative once, then $\mathcal{T}_{\mathcal{F}}$ halts immediately.
(2) Note that it would not be enough just to assume w.l.o.g. that all the sets of states of the members of $\mathcal{A}$ are pairwise disjoint and skip the "state relabeling process", since the flow-chart may contain several copies of the same machine, making it necessary to even take disjoint copies of one and the same machine (which is basically what we do when forming the machines $\mathbb{A}_i'$).

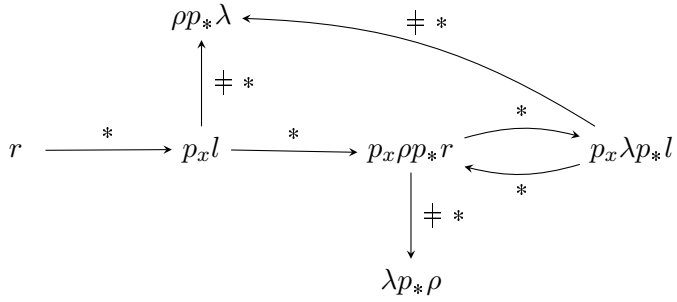
The power of this concept lies in its easy visualizability. We make the following conventions for drawing flow-charts: Before drawing, the alphabet X should always be fixed. We do not draw the vertices, but instead we write down names for the labels (so one Turing machine may appear at different positions in the drawing). The edges are, as usually, visualized by arrows with their labels noted next to them, but with the following additional rules for ease of notation:
(1) Whenever for some vertices $v, v' \in V$, for all $x \in X$ there is an edge $e \in E$ with $\iota(e) = v, \tau(e) = v'$ and $\lambda_E(e) = x$, in our visualization we only draw a single arrow without a label between the corresponding machine names. If there always is such an edge except for one particular $x_0 \in X$, then we also draw a single arrow, with label "$\neq x_0$".
(2) We may also just place several machine names next to each other, in a horizontal alignment, without any arrows, like: $\mathbb{A}_1 \cdots \mathbb{A}_n$. The meaning of this is defined recursively via $\mathbb{A}_1 \mathbb{A}_2 \cdots \mathbb{A}_n := \mathbb{A}_1 \to (\mathbb{A}_2 \cdots \mathbb{A}_n)$. Sometimes, we shall also write $\mathbb{A}^n$ for $\mathbb{A} \cdots \mathbb{A}$ (with n "factors").

(3) The machine representing the distinguished vertex will always be the leftmost in the drawing.

Example 1.2.10. This can be viewed as a continuation of Example 1.2.3. Now having the simple yet powerful concept of composition of Turing machines, we can build more complicated Turing machines out of the elementary ones. For ease of notation, let us fix the alphabet X as well as the number k of work tapes and $j \in \{1, \dots, k\}$, and write r for $r_X^{(j)}$ and l for $l_X^{(j)}$ as well as p_x for $p_X^{(j,x)}$. We also leave out the references to the fixed parameters in the defined names.
(1) The **(big) right machine** R moves the head to the first blank to the right of the consecutive sequence of (properly) imprinted cells into which it is placed at the beginning, and always at least one cell to the right:

(2) Analogously for the **(big) left machine** L, replacing r by l.
(3) Let $w = x_1 \cdots x_n \in X^*$. The **(big) printing machine w.r.t.** w, π_w, is defined by the following flow-chart: $p_{x_1} r p_{x_2} r \cdots p_{x_n}$.
(4) The **right search machine** ρ searches for imprinted cells to the right of its start position and halts on the first such cell (if there is none, then it does not halt):

(5) Analogously for the **left search machine** λ, replacing r by l.
(6) Fix any $x \in X_0$. The **search machine (w.r.t.** x**)** S halts on an imprinted cell (if there is none, it does not halt). In contrast to the left and right search machine, it will change the imprinting of the work tape in the course of its run, but at the end, the imprinting will be the same as in the start configuration; the symbol x is used as a marker during the run to show the boundary of the section of the work tape scanned so far:



As a consequence:

Proposition 1.2.11. *Let X be a finite alphabet with $0, 1 \in X_0$, $f : X_0^* \dashrightarrow X_0^*$. Then f is computable if and only if f is strongly computable.*

Proof. "⇐" is clear by definition, so we only show "⇒". Let $\mathbb{A}$ be a Turing machine over X computing f. Consider the left machine L and the search machine S operating on the first work tape. Then $SL\mathbb{A}$ strongly computes f: First, on any nonempty input $w \in X_0^*$, S halts on some cell imprinted with one of the symbols of w (because these are the only properly

imprinted cells on the tape). Then L moves the head to the first blank to the left of it, i.e., cell number 0, and the machine switches to the start state of $\mathbb{A}$, so we are in the usual start configuration of $\mathbb{A}$ on the given input, and an application of $\mathbb{A}$ will produce the desired output. □

From now on, instead of giving precise definitions of Turing machines used in our proofs, we will rather describe their functionality as explicitly as possible. From the formal point of view, these "proofs" should be viewed rather as extensive hints for "real" proofs, which would take too much space to write down. However, we encourage every reader sceptical about some part of such a proof to think about it with pencil and paper until they are convinced of the full formalizability of our argument. Furthermore, we will use the word "routine" to denote a Turing machine when viewed as a functional part of a bigger machine (by composition).
We now define the other two important and characteristic notions of recursion theory, beside computability:

Definition 1.2.12. *Let X be a finite alphabet containing the proper symbols 0 and 1.*
*(1) A subset $Q \subseteq X_0^*$ is called a **(decision) problem**.*
Now let $Q \subseteq X_0^*$ be a problem.
*(2) Q is called **(Turing-)recursively enumerable** if and only if $Q = \mathrm{ran}(f)$ for some computable $f : X_0^* \dashrightarrow X_0^*$.*
*(3) Q is called **(Turing)-recursive** or **(Turing-)decidable** if and only if both Q and $X_0^* \backslash Q$ are recursively enumerable.*

First note that in point (2) of Definition 1.2.12, we could replace the "$f : X_0^* \dashrightarrow X_0^*$" by "$f : X_0^* \to X_0^*$", and we would only lose $Q = \varnothing$:

Proposition 1.2.13. *Let X be a finite alphabet with $0, 1 \in X_0$, $\varnothing \neq Q \subseteq X_0^*$ a problem. If Q is recursively enumerable, then there exists a computable $g : X_0^* \to X_0^*$ with $\mathrm{ran}(g) = Q$.*

Proof. Let f be a partial function $X_0^* \dashrightarrow X_0^*$ such that $Q = \mathrm{ran}(f)$, and let $\mathbb{A}$ be a Turing machine over X computing f. Fix $w_0 \in Q$, and let $g := f \cup \{\langle w, w_0\rangle \mid w \in X_0^* \backslash \mathrm{dom}(f)\}$. Then $g : X_0^* \to X_0^*$ and $\mathrm{ran}(g) = \mathrm{ran}(f) = Q$. It remains to show that g is computable.
First, we show that we can assume w.l.o.g. that $\mathbb{A}$ always halts with the last head (corresponding to the "output tape") on cell number 0. If this is not the case, replace $\mathbb{A}$ by another Turing machine $\mathbb{A}'$ with one more work tape (and more states) than $\mathbb{A}$, which works as follows: On the "old" work tapes and in one of the "old" states, $\mathbb{A}'$ operates just as $\mathbb{A}$, also changing to the same old states, as long as they are different from $\mathbb{A}$'s halting state. On the new work tape, $\mathbb{A}'$ prints any proper symbol x on cell number 0 in its first work step, never changes the imprinting of the tape afterward, and always performs the same head movements as in the output tape, so the new head and the "output head" are always at the same position. Then, as soon as $\mathbb{A}$ would halt, $\mathbb{A}'$ instead switches to one of its new states, and starts running the standard search routine on the new tape with the output head following the movements of the new head, both eventually getting back to cell number 0 on their respective tape (since this is the only properly imprinted cell on the new tape). After this, $\mathbb{A}'$ halts, and exactly at the desired position, concluding the proof of this "w.l.o.g."-claim.
Now, assuming that $\mathbb{A}$ halts as desired, it is easy to check that the following Turing machine computes g: $\mathbb{A}r \xrightarrow{0} \pi_{1 \cdot w_0}$, where, of course, r and $\pi_{1 \cdot w_0}$ denote the corresponding standard routines for the output tape. □

The "w.l.o.g." argument in the proof of Proposition 1.2.13 is useful in many other situations as well. For example:

Proposition 1.2.14. *Let X be a finite alphabet with $0, 1 \in X_0$, $f, g : X_0^* \dashrightarrow X_0^*$ computable. Then the composition $g \circ f$ is computable.*

Proof. Note that by definition, $\mathrm{dom}(g \circ f) = f^{-1}(\mathrm{dom}(g)) \subseteq \mathrm{dom}(f)$. Let $\mathbb{A}$ be a Turing machine over X with k work tapes computing f, and $\mathbb{B}$ one with l work tapes computing g, w.l.o.g. assume that $\mathbb{A}$ always halts with the output head at position 0. We now describe a Turing machine $\mathbb{M}$ with $k + l$ work tapes computing $g \circ f$. On any input, $\mathbb{M}$ first simulates $\mathbb{A}$ on its first k work tapes. When $\mathbb{A}$ would halt, $\mathbb{M}$ moves the head on its k-th work tape one cell to the right and reads the imprinting there. If the imprinting is 0 (i.e., the input was not even an element of $\mathrm{dom}(f)$), it just moves the head on its last work tape one step to the right, prints 0 there and halts. Otherwise it moves both the head on the k-th and on the $(k + 1)$-th work tape one step to the right, then prints the "output" on the k-th work tape with the first bit deleted on the $(k + 1)$-th work tape and moves the head on this tape back to position 0 (by an application of L). After that, it simulates $\mathbb{B}$ on its last l work tapes and then halts. □

Definition 1.2.12 has the advantage of being easy to formulate, assuming the notion of computability is known. However, it might not be clear why these concepts are called like that; for this, we will need a characterization with "enumeration algorithms" and "decision algorithms":

Definition 1.2.15. *Let X be a finite alphabet with $0, 1 \in X_0$, $Q \subseteq X_0^*$ a problem.*
*(1) An **enumeration machine for** Q is a Turing machine $\mathbb{A}$ whose output head cannot move to the left and such that there exists a sequence $(w_n)_{n\in\omega}$ **enumerating** Q (i.e., for each $w \in Q$, there exists an $n \in \omega$ such that $w = w_n$) and such that $\mathbb{A}$ on the empty input does not halt, but for each $N \in \omega$ there exists an $m_0 \in \mathbb{N}$ such that after $m \geqslant m_0$ work steps of $\mathbb{A}$ on the empty input, the beginning of the imprinting of the positive section of the output tape of $\mathbb{A}$ reads as $w_0 * w_1 * w_2 \cdots w_N$, with the head on the output tape being to the right of the imprinting of w_N.*
*(2) A **decision machine for** Q is a Turing machine computing the characteristic function of Q in X_0^*.*

Theorem 1.2.16. *Let X be a finite alphabet with $0, 1 \in X_0$, $Q \subseteq X_0^*$ a problem.*
(1) If $Q \neq \varnothing$, then Q is recursively enumerable if and only if there exists an enumeration machine for Q.
(2) Q is decidable if and only if there exists a decision machine for Q.

Before proving Theorem 1.2.16, we need several lemmata, the first of which is a strengthening of the "w.l.o.g." observation from the proof of Proposition 1.2.13:

Lemma 1.2.17. *Let X be a finite alphabet, $\mathbb{A}$ a Turing machine over X. Then there exists a Turing machine $\mathbb{B}$ over X with at least two work tapes such that for all $w \in X_0^*$: $\mathbb{A}$ halts on w if and only if $\mathbb{B}$ halts on w, in which case the outputs of $\mathbb{A}$ and $\mathbb{B}$ on w are the same, and in the halting configuration of $\mathbb{B}$ on any input, all heads are at position 0 and the only imprintings on the tapes are the original input on the input tape and the output on the output tape; we say that $\mathbb{B}$ **halts nicely**.*

Proof. First, we may assume w.l.o.g. that the input and the output tape of $\mathbb{A}$ are distinct, since otherwise, $\mathbb{A}$ has only one tape, so just add another tape onto which, at the very beginning, we "copy" the input, then let both heads return to position 0 and finally simulate $\mathbb{A}$ on both tapes simultaneously.
The idea to construct $\mathbb{B}$ is similar to how we proved the "w.l.o.g." in Proposition 1.2.13. There, we used one additional "finding back" tape assigned to the output tape so that, in the end, we would find back to cell 0 on the output tape. Now, we will use one such "finding back" tape for each original tape of $\mathbb{A}$. But this will not be enough, because we also claim that, after having simulated $\mathbb{A}$ on an input w on which $\mathbb{A}$ halts, our machine $\mathbb{B}$ will "clean up", i.e., delete everything it imprinted except for the output on the output tape. For this, we will need to mark how far from cell 0 we got in each direction during the run. So we add two additional tapes, the "left boundary" and the "right boundary" tape. Also, note that the original input may be (partially) deleted during the run of $\mathbb{A}$, so we add one last tape, the "memory" tape.
In the very first work step of $\mathbb{B}$, any proper symbol x is imprinted on cell number 0 on each of the "auxiliary" tapes different from the "memory" tape (i.e., the "finding back" tapes and the "boundary" tapes), and then the input is copied to the "memory" tape by moving the input and the input memory head. After that, both these heads return to position 0, and the machine starts simulating $\mathbb{A}$ on w on the non-auxiliary tapes.
During this, the "finding back" heads follow the movements of the head they are assigned to, and the "left boundary" head always moves one cell to the left, whereas the "right boundary" head keeps moving to the right. The memory head stays where it is.
If $\mathbb{A}$ does not halt on w, then of course the simulation does not halt either, so $\mathbb{B}$ does not halt. However, if $\mathbb{A}$ halts on w, then after the simulation of $\mathbb{A}$, $\mathbb{B}$ first moves all its heads except for the two boundary heads and the memory head back to position 0 on their corresponding tape. Then, $\mathbb{A}$ starts the "left cleaning routine": It keeps moving all non-boundary and non-memory heads to the left, replacing any proper symbol occurring by $*$, while moving the left boundary head to the right and the keeping the right boundary and the memory head where they are. This goes on until the left boundary head reaches position 0. Note that also the "finding back" heads again move along with the $\mathbb{A}$ heads. Now the tapes are all "cleaned up" at the negative positions, so $\mathbb{B}$ lets its "$\mathbb{A}$ heads" and the "finding back" heads return to position 0.
Of course, the "right cleaning routine" will be similar, but one needs to make sure the output is not deleted on the output tape, which is not a problem. Finally, when all heads have returned to position 0, the input is copied from the memory tape back to the input tape deleting the memory tape's imprinting, moving the input, the memory, and the "finding back" head assigned to the input tape repeatedly to the right. Then the three heads are moved back to position 0, all markers x on the auxiliary tapes are deleted, and $\mathbb{B}$ halts. □

The proof of the next lemma is not difficult, and we leave it as an exercise:

Lemma 1.2.18. *Let X be a finite alphabet with $0, 1 \in X_0$, $<$ a linear order on X_0. Let $\lhd$ denote the lexicographical order on X_0^* induced by $<$; note that $\lhd$ is a well-order of type ω, so in particular, each $w \in X_0^*$ has a unique successor under $\lhd$. Then the functions $\Sigma : X_0^* \to X_0^*$, assigning to each word in X_0^* its successor under $\lhd$, and $\nu : X_0^* \to X_0^*$, assigning to each word $w \in X_0^*$ the word $1^{n(w)}$, where $n(w)$ is defined recursively via $n(\varnothing) = 0$ and $n(\Sigma(w)) = n(w) + 1$, are computable.* □

Proof of Theorem 1.2.16. For (1), "⇒":
By Proposition 1.2.13, there exists a computable $f : X_0^* \to X_0^*$ such that $\mathrm{ran}(f) = Q$. Fix a nicely halting Turing machine $\mathbb{A}$ on X computing f, say with k work tapes. Also, fix a linear order $<$ on X_0 and a nicely halting Turing machine $\mathbb{O}$, say with l work tapes, computing Σ as in Lemma 1.2.18. We now build out of $\mathbb{A}$ and $\mathbb{O}$ an enumeration machine $\mathbb{E}$ for Q.
$\mathbb{E}$ has $k + l$ work tapes, with the first k work tapes corresponding to the work tapes of $\mathbb{A}$, the first and the $k + 1$-th to $k + l - 1$-th work tape corresponding to $\mathbb{O}$'s work tapes, and the last work tape being the "enumeration tape". On any input, $\mathbb{E}$ starts by simulating $\mathbb{A}$ on the corresponding tapes. After that, the output of $\mathbb{A}$ with the first bit (1) removed is copied to the enumeration tape. Then the head on the enumeration tape moves one more step to the right, and the output on the k-th tape is deleted, with the output head returning to position 0.
After this, $\mathbb{E}$ simulates $\mathbb{O}$ on the corresponding tapes, and after the simulation is finished, copies the output on its second-to-last tape (the successor of the input) to its first tape, overwriting the original input and also deleting the output on the second-to-last tape and moving all heads except for the enumeration head back to position 0. After that, $\mathbb{E}$ simulates $\mathbb{A}$ with the increased input again, and so on. Note that during all these work steps, the enumeration head does not move to the left.
Now, since $\varnothing$ is the $\lhd$-smallest element of X_0^*, on it as input, $\mathbb{E}$ will successively enumerate all elements of $\mathrm{ran}(f)$ on its enumeration tape.
For (1), "⇐":
Fix an enumeration machine $\mathbb{E}$ for Q with k work tapes, as well as a nicely halting Turing machine $\mathbb{O}$ computing ν as in Lemma 1.2.18, with l work tapes. We now describe how to build a Turing machine $\mathbb{A}$ with $k + l + 3$ work tapes computing some function $f : X_0^* \to X_0^*$ with $\mathrm{ran}(f) = Q$.
The first l work tapes of $\mathbb{A}$ correspond to those of $\mathbb{O}$, the next k tapes to those of $\mathbb{E}$. We also add two "finding back" tapes whose functionality we will describe later as well as a new output tape.
The idea is: For any $w \in X_0^*$, $f(w)$ should be the $n(w)$-th word listed by $\mathbb{E}$, with n as in Lemma 1.2.18. So, on any input w, $\mathbb{A}$ first runs $\mathbb{O}$ on its first l work tapes to get a chain of 1's serving as a "counter".
After this, it prints a marker x on cell number 0 on the first "finding back" tape, and starts running $\mathbb{E}$ on tapes $l + 1$ to $l + k$. The two "finding back" heads follow the head on the $(l + k)$-th tape. After each work step of $\mathbb{E}$, $\mathbb{A}$ interrupts, marks the current position of the $(l + k)$-th tape's head by x on the second "finding back" tape, and moves the head on the $(l + k)$-th tape as well as its two "finding back" heads back to position 0. Then it reads what $\mathbb{E}$ has printed so far, counting by moving the head on the l-th tape along the chain of 1's.
Note that the last word on the enumeration tape of $\mathbb{E}$ may only have been imprinted partially yet, so only words followed by $*y$, where y is any proper symbol, count. Anyway, if the number of listed words is not large enough yet, the counter head moves back to the beginning, and via the second "finding back" head, the $(l + k)$-th head and the two "finding back" heads return to where the output head of $\mathbb{E}$ was after its last work step. Then the marker on the second "finding back" tape is deleted, and $\mathbb{A}$ lets $\mathbb{E}$ carry out its next work step.
Finally, when enough words have been listed by $\mathbb{E}$, $\mathbb{A}$ copies the corresponding word from the list, preceded by one bit 1, to its output tape, and halts.
For (2), first observe that this equivalence is clear when $Q \in \{\varnothing, X_0^*\}$, so we can focus on nonempty problems Q with nonempty complement in X_0^*.

For (2), "⇒":
This is similar to the proof of (1), "⇐" we just gave: If Q is decidable, then by the already established equivalence (1), there exist enumeration machines $\mathbb{E}_1$ and $\mathbb{E}_2$ for Q and $X_0^* \backslash Q$ respectively. Place the two machines atop of each other, add two "finding back" tapes for each output tape of the $\mathbb{E}_i$, and one input tape at the very beginning as well as one output tape at the very end. Alternately, run $\mathbb{E}_1$ and $\mathbb{E}_2$ for one work step and check whether the input already appears in one of the lists. It must eventually appear in precisely one of them, and then $\mathbb{A}$ answers accordingly.
For (2), "⇐":
Fix nicely halting Turing machines $\mathbb{A}$ computing the characteristic function of Q in X_0^*, say with k work tapes, and $\mathbb{B}$, computing Σ as in Lemma 1.2.18, say with l work tapes. W.l.o.g., we only show how to produce an enumeration machine $\mathbb{E}$ for Q.
$\mathbb{E}$ is obtained by placing $\mathbb{A}$ and $\mathbb{B}$ atop of each other, identifying their input tapes and adding a new "enumeration" tape at the end (so $\mathbb{E}$ has $k + l$ work tapes).
On any input w, $\mathbb{E}$ first runs $\mathbb{A}$ to get either 10 or 11 as output, depending on whether $w \notin Q$ or $w \in Q$. In the first case, some fixed string $w_0 \in Q$ is printed on the enumeration tape, whereas in the second case, the input is printed there. In both cases, the output 10 or 11 is deleted, and $\mathbb{B}$ is run to get the next input to check, but on the $(k + l)$-th tape, so it will be copied to the first tape, overwriting the input there and deleting the content of the $(k + l)$-th tape simultaneously. Then this process starts anew. Clearly, $\mathbb{E}$ is an enumeration machine for Q. □

The following proposition gives another characterization of recursively enumerable sets which we will use in the proof of Matiyasevich's Theorem. We leave the proof as an exercise; note that before doing this exercise, you may find it useful to take a look at Example 1.3.13:

Proposition 1.2.19. *Let X be a finite alphabet with $0, 1 \in X_0$, $Q \subseteq X_0^*$ a problem. Then Q is recursively enumerable if and only if there exists a Turing machine over X that halts precisely on those inputs from X_0^* which are in Q.* □

We end this subchapter with an important concept for deriving the (un-)decidability of some set of words from the according property of another:

Definition 1.2.20. *Let X, Y be finite alphabets, $Q_1 \subseteq X$ and $Q_2 \subseteq Y$. A function $f : X_0^* \to Y_0^*$ is called a **reduction from** Q_1 **to** Q_2 if and only if:*
(1) f is computable as a partial function $(X \cup Y)_0^* \dashrightarrow (X \cup Y)_0^*$ and
(2) $\forall w \in X_0^* : w \in Q_1 \Leftrightarrow f(w) \in Q_2$.

The following lemma illustrates the idea behind reductions; we leave its proof as an exercise in constructing Turing machines (you may use Lemma 1.3.3 from the next subchapter in your argumentation):

Lemma 1.2.21. *Let X, Y be finite alphabets with $0, 1 \in X_0 \cap Y_0$, $Q_1 \subseteq X_0$, $Q_2 \subseteq Y_0$, and suppose there exists a reduction from Q_1 to Q_2. Then: If Q_2 is decidable, so is Q_1.* □

Of course, for us, the contraposition of the statement from Lemma 1.2.21 will be important: Reducing from an undecidable set gives another undecidable set.

1.3 Universal Turing Machines and Undecidability

Our goal in this subchapter, which is also based on (8) and (12), with the proof of Theorem 1.3.12 inspired by the one of Theorem 7.5 in (10) on page 219, is to show that for any finite alphabet X with $0, 1 \in X_0$, there exists a recursively enumerable, but not recursive subset $Q \subseteq X_0^*$. Note that the existence of not recursive subsets alone follows from a simple cardinality argument, since there are $2^{\aleph_0}$ many subsets of X_0^* altogether, but only $\aleph_0$ of them are recursive. However, since there are also only $\aleph_0$ many recursively enumerable subsets, we cannot apply this argument here.
Actually, we first need the existence of so-called "universal Turing machines". For this purpose, it is useful to know that, if we are only concerned with computation power (not efficiency), we can restrict ourselves to 1-tape Turing machines:

Theorem 1.3.1. *Let X be a finite alphabet, $\mathbb{A}$ a Turing machine over X. Then there exists a Turing machine $\mathbb{B}$ over X with only one work tape such that for any input $w \in X_0^*$, $\mathbb{A}$ halts on w if and only if $\mathbb{B}$ does, in which case the outputs of $\mathbb{A}$ and $\mathbb{B}$ on w are the same and $\mathbb{B}$ **halts nicely**, i.e., on cell number 0 and with the output the only imprinting on the work tape.*

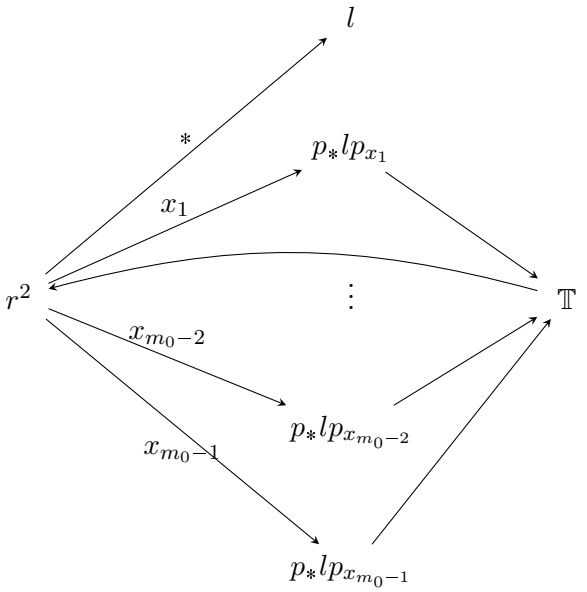
Remark 1.3.2. Note that before Theorem 1.3.1, we had only defined the notion of "halting nicely" for Turing machines with at least two work tapes, so this completes the definition for all Turing machines.

As for Theorem 1.2.16, we will do some of the proof work outside of the actual proof by proving a lemma:

Lemma 1.3.3. *Let X, Y be finite alphabets with $X \subseteq Y$ and $0, 1 \in X_0$. Furthermore, let $\mathbb{A}$ be a one-tape Turing machine over Y such that for all $w \in X_0^*$: If $\mathbb{A}$ halts on input w, then the output is in X_0^*. Then there exists a nicely halting one-tape Turing machine $\mathbb{B}$ over X such that for all $w \in X_0^*$, $\mathbb{B}$ halts on w if and only if $\mathbb{A}$ does so, with the same output as $\mathbb{A}$.*

Proof. Since, in order to build $\mathbb{B}$, we have less symbols at our disposal, we will need to do some encoding. First, say $|X| = m_0$ and $|Y| = m_1$. Note that $m_0 \geqslant 3$, so we can choose $k \in \mathbb{N}$, $k \geqslant 2$, so big that $(m_0 - 1)^{k-1} \geqslant m_1$.
Now we want to encode all symbols from Y ($*$ as well) into $(k-1)$-tuples of symbols from X such that none of these tuples contains $*$. This is possible by the choice of k; fix such an encoding. The actual encoding on tape will be by strings of length k, where the first $k-1$ entries form one of the encodings we just fixed, and the last bit is either 0 or 1, depending on whether the string stands for the imprinting of cell number 0 or another cell. We need this to "find back" at the end (note that our standard "trick" of finding back does not work here, since we are not allowed to add other tapes). We think of the cells of $\mathbb{B}$'s work tape as grouped together such that the imprinting of cells number $(n-1) \cdot k + 1, \ldots, n \cdot k$ encodes the imprinting of cell number n on the tape of $\mathbb{A}$ during the simulation of $\mathbb{A}$ by $\mathbb{B}$.
We now describe what $\mathbb{B}$ does on any input $w \in X_0^*$. First, $\mathbb{B}$ prints the encoding for $*$ plus last bit 0 on its cells number $-k+1, \ldots, 0$, returning then to cell number 0. Now note that the input w is handed over to $\mathbb{B}$ in unencoded form, so we cannot start with the simulation right away, but need to make $\mathbb{B}$ encode the input first.
First, $\mathbb{B}$ checks whether w is empty or not by looking at cell number 1. If $w = \varnothing$, then $\mathbb{B}$ returns to cell 0 and starts the simulation described later immediately. Otherwise, it deletes the first symbol of the input, memorizing it by changing to a certain state, goes to the end of

the input, passes over one $*$, and then prints the encoding for the imprinting of the symbol it memorized (with 1 as the last bit, since this will stand for the imprinting of cell 1). It continues to encode the original input, going back and forth. Note that as long as the original input has not been deleted completely, the space to the left of the encoding string will always be of the form $x*$, where x is a proper symbol (the last original input symbol), and afterward of the form $**$, making it possible for $\mathbb{B}$ to recognize when this routine is finished.
Now the input has been printed in encoded form, but there is a big gap of blanks between it and cell number 0, so $\mathbb{B}$ next must move the encoded input repeatedly to the left until its first symbol is at position 1. For this, it applies the so-called **left translation routine**, which, when started two cells to the left from a string W of proper symbols separated by a blank on each side, will move W one step to the left and halt at the new $*$ to the right of W. It is given by the following flow-chart, where $X = \{*, x_1, \ldots, x_{m_0-1}\}$ and $\mathbb{T}$ is the **trivial machine**, which goes from the start state straight to the halting state without doing anything; it was added here only to make the drawing nicer:



How does $\mathbb{B}$ know when to stop moving the encoded input string to the left? For this, after each application of the left translation routine, $\mathbb{B}$ will start moving its head to the left, keeping track of how many steps modulo k it took. After its first step to the left (stepping on the last bit of the encoding), it will check every k steps whether the imprinting is 1 (meaning it is still on the string and needs to keep moving), 0 (meaning the string has been moved where it should be and the routine can stop) or $*$ (meaning it has reached the end of the encoding and needs to apply the translation routine again).
After all this, the simulation of $\mathbb{A}$ can finally start. How this is carried out should be clear by now, but we note one thing: Whenever $\mathbb{B}$ steps over the section of the tape it has worked on so far (which it can recognize by reading $*$; this is why we wanted no encoding to contain $*$), it will always imprint these next k cells with some encoding. This way, the entire tape section worked on so far will always be covered by proper symbols.
Now, after the end of the simulation in case of an input w that $\mathbb{A}$ halts on, $\mathbb{B}$ still needs to "clean up" and reconvert the output into its usual form. First, it goes to the left end of the work section (which is easy by the remark in the last paragraph). Then it starts moving

right again, keeping track of how many steps modulo k it took and deleting everything that it passes over, looking at the imprinting of the "relevant" cells (i.e., the cells at the end of each k-block) to eventually know when it has reached cell number 0. The cells at negative positions are all "cleaned up" by then, but we leave the symbol 0 in cell number 0 standing for the moment in order to find back there.
Next, the head passes over the output. It will know it is at the first cell after the output when it either reads $*$ (in case the end of the output is already the right end of the work section) or the encoding for $*$. In the second case, it "cleans up" to the right of the output before starting to reconvert the output, whereas in the first case, it starts with the reconversion right away.
As for this final routine, it should suffice to mention that the first cell to the right of the encoded output will be marked by any proper symbol, and then the head returns to cell number 0 to reconvert the output from the left to the right. The machine will know when the reconversion is complete since the right marker cell will be the only imprinted cell left from the string to be reconverted, but $k \geqslant 2$. Finally, $\mathbb{B}$ returns its head to the left end of the properly imprinted cells, i.e., cell number 0, deletes the 0 there, and halts. □

Remark 1.3.4. Note that also the case $Y = X$ in Lemma 1.3.3 gives an interesting result: We cannot only w.l.o.g. assume "nice halting" for multi-tape Turing machines, but also for one-tape machines.

Proof of Theorem 1.3.1. By Lemma 1.3.3, it is sufficient to find a one-tape Turing machine $\mathbb{B}$ as required, but over a bigger finite alphabet Y and which does not need to halt nicely. The idea is: Fuse all work tapes of $\mathbb{A}$ into a single work tape, making up for the fact that each cell now stands for a tuple of cells with a tuple of imprintings by using more symbols. However, there is another obstacle to overcome: There is only one head now, but the heads of $\mathbb{A}$ need not always be at the same position.
Let k be the number of work tapes of $\mathbb{A}$ and $m_0 = |X|$. We then fix any superset Y of X with $|Y| = 2^{k+1} \cdot m_0^k + 1$. We view Y_0 as partitioned into $2 \cdot 2^k$ clusters of m_0^k symbols, where each subset of the set of heads of $\mathbb{A}$ corresponds to precisely two clusters. This enables us to use the proper symbols of Y not only to denote entire k-tuples of symbols of X, but also which heads are currently at that position, and we always have a second symbol with this meaning, which we only use for cell number 0 (again, for "finding back").
With these ideas for the construction of $\mathbb{B}$ given, we leave the details as an exercise; as a hint: The construction is actually quite similar to the one in the proof of Lemma 1.3.3. During the simulation of $\mathbb{A}$, make sure in each work step simulation, the positions of the heads of $\mathbb{A}$ are visited one after another to "do their work". At the end, $\mathbb{B}$ must reconvert at least the output and the first blank after it cell-wise, throwing away the information about the imprinting of the cells of the other tapes at these positions. □

Universal Turing machines are Turing machines which can simulate any Turing machine; by Lemma 1.3.3 and Theorem 1.3.1, we do not really lose the "universality" when we restrict ourselves to single-tape machines over the alphabet $\{*, 0, 1\}$. So, we want a Turing machine (n.n. single-tape, but this can then always be achieved by Theorem 1.3.1) which, given a description of a single-tape machine over $\{*, 0, 1\}$, a configuration of this machine and a natural number k, computes the k-th configuration of the described machine after the given one. The construction of such a machine itself is not difficult, once we have suitably formalized the problem, which involves some work.

First, it looks as if we are not only handing over one input to the universal machine, but three inputs at once. Formally, however, by our definition, Turing machines only accept one string as input. We could generalize this definition by allowing several strings to be printed on the input tape, each separated by one $*$. But we can also just define the following:

Definition 1.3.5. *Let X be a finite alphabet with $0, 1 \in X_0$.*
(1) For each $k \in \mathbb{N}, k \geqslant 1$ and $w \in X_0^*$, let $w^{(k)}$ denote w with each letter x replaced by x^k (e.g., $01^{(2)} = 0011$).
*(2) For $w_1, \ldots, w_n \in X_0^*$, the **encoding n-tuple of** $w_1, \ldots, w_n$ is defined as $[w_1, \ldots, w_n] := w_1^{(2)}01w_2^{(2)}01 \cdots w_n^{(2)}$.*

As an easy exercise:

Proposition 1.3.6. *Let X be a finite alphabet with $0, 1 \in X_0$.*
*(1) Encoding tuples over X possess **unique readability**, i.e.: For any $v \in X_0^*$ such that v is an encoding tuple over X, there exists a unique $n \in \mathbb{N}$ and unique $w_1, \ldots, w_n \in X_0^*$ such that $v = [w_1, \ldots, w_n]$.*
*(2) The partial functions $\lambda : X_0^* \dashrightarrow X_0^*$ assigning to each encoding tuple the string 1^l, l the **size** of the tuple (i.e., the number of components) and $\pi_k : X_0^* \dashrightarrow X_0^*$ for $k \in \mathbb{N}, k \geqslant 1$, assigning to each encoding tuple of size at least k its k-th component, are computable.* □

So, instead of handing over several separated strings as input, we form their encoding tuple, which is a single string, and hand this over as input. By Proposition 1.3.6, the machine can always split it up into its components.
Now, there remains the question how to encode single-tape Turing machines, configurations and natural numbers as strings over $\{0, 1\}$. As for natural numbers, we choose the standard **binary representation**, where e.g., 5 is coded as 101, and 0 as itself. We write $\mathrm{bin}(n)$ for the binary representation of n.
As for Turing machines $\mathbb{A}$: We first choose encodings for $\mathbb{A}$'s states and the three move types $-1, 0$ and $+1$ as strings over $\{0, 1\}$. We encode all states of $\mathbb{A}$, which are literally natural numbers, by their binary representation, and the move -1 by 01, the move 0 by 00 and the move $+1$ by 10. Also, since we do not want to use the improper symbol $*$ in the encoding, we use encodings for each occurrence of one of the letters $*, 0$ and 1 in the matrix: $*$ is encoded by 00, 0 by 10 and 1 by 11.
Then we can just encode the matrix that $\mathbb{A}$ literally is rowwise as a large tuple; we do not need to form a "tuple of tuples", since we know that these matrices always have five columns, so as long as the universal machine keeps track of at which entry modulo 5 it is, it will know the column number. Note however, that by our Definition 1.2.1, the halting state of $\mathbb{A}$ may never be mentioned at all in the whole matrix; for completeness, we add it as the very last entry of the tuple, regardless of whether it is mentioned or not. So, just to check: When $\mathbb{A}$ has s different states, then it will be encoded by a tuple of size $15 \cdot (s - 1) + 1$, which we denote by $\ulcorner \mathbb{A} \urcorner$ and call the **Gödelization of** $\mathbb{A}$.
As for configurations: Note that these are not essentially finitistic objects by definition. However, since the input for the universal machine includes a bound k on the number of work steps to be carried out, it suffices to encode a section of length $2k + 1$ on the work tape, where the initial head position is in the middle. On the one hand, we do not want to use the improper symbol $*$ to encode empty cells in the configuration, on the other hand, we also want to mark the current head position (together with the imprinting of the cell) in the

encoding, so we use three bits per cell to be encoded and write 000 for "$*$, no head", 010 for "0, no head", 011 for "1, no head", and the same with the first bit changed to 1 for the cell carrying the head.
So, the "official definition" is as follows: A **configuration encoding of dimension k over** S, where $S \subseteq \mathbb{N}$ is finite, is a string of the form $[\mathrm{bin}(s), t]$, where $s \in S$ and t is some concatenation of $2k+1$ strings each of the form 000, 010, 011, 100, 110 and 111, where one of the last three strings appears precisely once in the concatenation, and the other two do not appear at all. If the string with first bit 1 appears precisely in the middle, the encoding is called **centered**.
Fix $k \in \mathbb{N}, k \geqslant 1$. Note that to each configuration encoding of dimension k over S, we can assign a real configuration over S in a canonical way, where the head position is cell 0 and the cells of distance more than k from it are all empty. Conversely, for each configuration over S and each $n \in \mathbb{Z}$ such that the head of the configuration is at one of the positions $\{n-k, n-k+1, \ldots, n+k\}$, we can assign in a canonical way a configuration encoding of dimension k representing the tape section from cells $n-k$ to $n+k$ under this configuration. We can now finally define:

Definition 1.3.7. *Let $X := \{*, 0, 1\}$.*
*(1) The partial function $\mathcal{S} : X_0^* \dashrightarrow X_0^*$, called the **universal single-tape simulation function**, is defined as follows: Its domain is the set of all strings of the form $[\ulcorner \mathbb{A} \urcorner, \mathrm{bin}(k), c]$, where $\mathbb{A}$ is a single-tape Turing machine with state set $S \subseteq \mathbb{N}$ over X, $k \in \mathbb{N}$ and c a centered configuration encoding of dimension k over S. $\mathcal{S}$ maps such a string to c', where c' is the canonical configuration encoding of dimension k over S representing the tape cells number $-k$ to k of the k-th successor configuration under $\mathbb{A}$ of the canonical configuration over S associated to c.*
*(2) A **universal single-tape Turing machine** is any single-tape Turing machine over X computing the universal single-tape simulation function.*

We will use the following two lemmata, the proof of which is left as an exercise, in the proof of the existence of universal single-tape Turing machines:

Lemma 1.3.8. *The domain of $\mathcal{S}$ as in Definition 1.3.7(1) is decidable.* □

Lemma 1.3.9. *Let $X := \{*, 0, 1\}$. The partial function $f : X_0^* \dashrightarrow X_0^*$, mapping each string of the form $\mathrm{bin}(n)$ for some $n \in \mathbb{N}$ to the string 1^n, is computable.* □

Theorem 1.3.10. *There exists a universal single-tape Turing machine.*

Proof. Note that by Theorem 1.3.1, it suffices to construct any Turing machine $\mathbb{U}$ (n.n. single-tape) over the alphabet $\{*, 0, 1\}$ computing $\mathcal{S}$ as in Definition 1.3.7(1).
But this is easy, using three tapes, where the second one is referred to as the "counter tape". At the very beginning, $\mathbb{U}$ checks whether the input is in the domain of $\mathcal{S}$, which is possible by Lemma 1.3.8. To be more precise: The input is copied to the output tape, and then any nicely halting one-tape routine computing the characteristic function of $\mathrm{dom}(\mathcal{S})$ is run on that tape. The result is either 10 or 11. In the first case, this string is replaced by a single 0, and $\mathbb{U}$ halts. In the second case, all two bits are deleted, a 1 is imprinted on cell number 1 on the output tape, and $\mathbb{U}$ goes on as follows:
It moves the input head to the second entry $\mathrm{bin}(k)$ of the triple $[\mathbb{A}, \mathrm{bin}(k), c]$ and prints it starting at cell number 1 on the counter tape. Then it runs a single-tape nicely halting routine

computing the function f from Lemma 1.3.8 on the counter tape, deleting the first bit 1 of the output 1^{k+1} and moving it one cell to the left by an application of the left translation routine afterward. Then the counter head moves to the end of the now imprinted string 1^k. After this, the last entry c from the input is printed on the output tape, starting at cell number 2. Now $\mathbb{U}$ starts simulating the work of $\mathbb{A}$ on the configuration encoding on the output tape, searching the input for the unique matching assignment and, after each work step simulation, moving the counter head one step to the left. Should the halting state of $\mathbb{A}$ be reached before the counter head has arrived at the first blank cell (which can also be checked after each step by looking at the very last entry of $\ulcorner\mathbb{A}\urcorner$), no further manipulations on the output tape are carried out, and $\mathbb{U}$ halts immediately. □

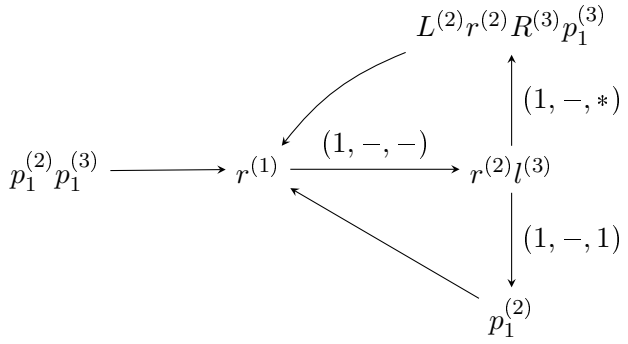
Remark 1.3.11. Our Definition 1.3.7(2) is not the most general possible. Assuming w.l.o.g. that the alphabets used by Turing machines are all finite subsets of $\mathbb{N}$, we can encode all letters used by any Turing machine by their binary representations. Tuples formed with these letters can be encoded by encoding tuples. Then it is not difficult to define canonical encodings for all Turing machines and configurations (not just single-tape ones) as words over $\{0,1\}$. Analogously to Theorem 1.3.10, one can prove the existence of a single-tape Turing machine over $\{*,0,1\}$ simulating any other Turing machine; see (8), pp.203ff., for a proof of at least this statement with "any other Turing machine" replaced by the weaker "any other single-tape Turing machine (n.n. over $\{*,0,1\}$)". We will not need any of the more general results in this thesis, though.

Our final theorem in this subchapter, as already mentioned at the beginning:

Theorem 1.3.12. *Let X be a finite alphabet with $0,1 \in X_0$. Then there exists a recursively enumerable but not recursive subset $Q \subseteq X_0^*$.*

Note that this is the first negative computability result that we will prove in this thesis. It is a source for many other algorithmic undecidability instances; for example, in Chapter 4, we will be able to build from this finite group presentations with undecidable word problem.
Before proving Theorem 1.3.12 with the help of Theorem 1.3.10, we need an important ingredient:

Example 1.3.13. We endow the set $\mathbb{N}^2$ with the following relation $\lhd$: $(x_1,y_1) \lhd (x_2,y_2)$ if and only if $x_1+y_1 < x_2+y_2 \vee (x_1+y_1 = x_2+y_2 \wedge x_1 > x_2)$. It is easy to check that $\lhd$ is a well-order of type ω on $\mathbb{N}^2$ (via which $\mathbb{N}^2$ is enumerated "diagonalwise"). Let $\varphi : \omega \to \mathbb{N}^2$ be the unique order isomorphism. For example, $\{\varphi(0),\ldots,\varphi(4)\} = \{(0,0),(1,0),(1,1),(2,0),(2,1)\}$.
Now, consider the three-tape TM $\mathbb{D}$ given by the flow-chart depicted at the end of this example (the exponents in brackets denote the tape where the corresponding routine is run, and $-$ in the triples stands for an arbitrary symbol).
It is readily checked that $\mathbb{D}$, on an input of the form 1^n, $n \in \mathbb{N}$, halts with the second and third head at nonnegative positions x and y respectively, where $\varphi(n) = (x,y)$, and with the imprinting of the second and third tape being 1^{k+1} and 1^{l+1} respectively, where k and l are respectively the maximum first and second coordinate among the pairs $\varphi(0),\ldots,\varphi(n)$. We call $\mathbb{D}$ the **diagonal machine**.



Finally, another lemma, which is immediate from Lemma 1.3.8:

Lemma 1.3.14. *Let $X := \{*, 0, 1\}$. Then the set of all Gödelizations of single-tape Turing machines over X is a decidable and thus recursively enumerable subset of X_0^*.* □

Proof of Theorem 1.3.12. Let $Q := \{\ulcorner\mathbb{A}\urcorner \mid \mathbb{A} \text{ is a single-tape TM over } \{*, 0, 1\} \text{ halting on } \ulcorner\mathbb{A}\urcorner \text{ with output } 0\} \subseteq \{0, 1\}^* \subseteq X_0^*$. We claim that Q is recursively enumerable, but not recursive.

First, suppose Q was recursive. By Theorems 1.2.16 and 1.3.1 as well as Lemma 1.3.3, we can fix a single-tape Turing machine $\mathbb{A}$ over $\{*, 0, 1\}$ computing the characteristic function of Q in X_0^*. From $\mathbb{A}$, we easily get a single-tape TM $\mathbb{B}$ over $\{*, 0, 1\}$ such that on any input $w \in X_0^*$, $\mathbb{B}$ outputs 1 if and only if $w \in Q$, and 0 otherwise.

But then we have the following equivalence: $\ulcorner\mathbb{B}\urcorner \in Q \overset{\text{Def. of } Q}{\Longleftrightarrow} \mathbb{B}$ halts on $\ulcorner\mathbb{B}\urcorner$ with output $0 \overset{\text{Choice of } \mathbb{B}}{\Longleftrightarrow} \ulcorner\mathbb{B}\urcorner \notin Q$, a contradiction.

It remains to show that Q is recursively enumerable. The idea is as follows: Find a Turing machine $\mathbb{M}$ which, for each single-tape Turing machine $\mathbb{A}$ over $\{*, 0, 1\}$ and each $k \in \mathbb{N}$, eventually checks whether $\mathbb{A}$ halts on $\ulcorner\mathbb{A}\urcorner$ within at most k steps with output 0 and, if then so, lists $\ulcorner\mathbb{A}\urcorner$. Obviously, we will need a nicely halting universal single-tape Turing machine $\mathbb{U}$ here, and we will use the diagonal routine $\mathbb{D}$ from Example 1.3.13 to go through all cases to check in an appropriate order. By Lemma 1.3.14, we can also fix an enumeration machine $\mathbb{E}$ for the set of all Gödelizations of one-tape Turing machines over $\{*, 0, 1\}$.

We get $\mathbb{M}$ by placing $\mathbb{D}$, $\mathbb{E}$, two "finding back" tapes for the enumeration tape of $\mathbb{E}$ as in the proof of "(1),⇐" of Theorem 1.2.16, $\mathbb{U}$ and one final tape, the enumeration tape of $\mathbb{M}$, atop of each other.

The very first tape of $\mathbb{M}$ (i.e., the first tape of $\mathbb{D}$) serves as a counter tape. It will be imprinted with a string of 1's to which, after each check as described above is complete, another 1 is added at the end. In the beginning this tape is empty (recall that we need to control the behavior of $\mathbb{M}$ on the empty input), i.e., imprinted with the string 1^0.

The checking steps of $\mathbb{M}$ go as follows: First, $\mathbb{D}$ is run to get, on the second tape, the number (according to the list created by $\mathbb{E}$) of the Turing machine $\mathbb{A}$ to be checked and, on the third tape, the number of steps to be checked for (deleting the 1s to the right of the two positions to find back). It is then checked (with the help of the first finding back tape) whether $\mathbb{E}$ has already imprinted enough Gödelizations on its enumeration tape. If not, then with the help of the second finding back tape, $\mathbb{E}$ is run for another step and the check is repeated.

Once this check is passed, an according input is printed on the tape of $\mathbb{U}$, the strings of 1s on the second and third tape of $\mathbb{D}$ are deleted, and then $\mathbb{U}$ is run. If it is accepting (i.e.,

the output starts with 1), it is checked together with the imprinting on $\mathbb{E}$'s enumeration tape whether the halting state in the configuration encoding was reached and, if so, whether the output is 0. If in at least one of these two checks, the answer is "no", the imprinting on the $\mathbb{U}$ tape is deleted immediately, the counter on the very first tape is raised by 1, and the next check is initiated.
Otherwise, the imprinting of the $\mathbb{U}$ tape is deleted as well, but before raising the counter and going on to the next step, the according Gödelization from the enumeration tape of $\mathbb{E}$ is copied to the enumeration tape of $\mathbb{M}$. □

The proof of Theorem 1.3.12 also shows that the set of all encoding pairs $[\ulcorner\mathbb{A}\urcorner, w]$, where $\mathbb{A}$ is a single-tape TM over $\{0, 1, *\}$ halting on $w \in \{0, 1\}^*$ is undecidable; this set is known as the **halting problem**. To see that it is undecidable, just note that if we could decide the halting problem algorithmically, we could also easily decide the set Q from the proof; given any input, first check whether it is a Gödelization of a single-tape Turing machine over $\{0, 1, *\}$; if not, reject. Else check, using the decidability of the halting problem, whether the Turing machine encoded by the input halts on its own Gödelization; if not, reject. Else simulate the machine using a universal Turing machine until it halts and check whether the output is 0. If not, reject, else accept.

1.4 One-sided tape Turing machines

This is a different, but "equivalent" notion of Turing machines that will make notation (more precisely: the coding of configurations) in the proof of Matiyasevich's Theorem 2.1.4 a bit easier; since we now already assume some familiarity with the formalism behind Turing machines, we do not give the full formal definition, but rather describe the intuition, following the exposition in (11) and (12).
A **Turing machine with one-sided tapes** is defined similarly to the Turing machines that we have worked with so far, but the work tapes are not infinite in both directions, but have a left end. Thus cells on a work tape now do not correspond to integers, but to natural numbers. The leftmost cell of such a work tape carries a special symbol, which we will always denote by §, and which tells the machine that it must not move the corresponding head further to the left when it is over that cell. Of course, § must never be overwritten by another symbol, and it must not be printed on cells different from cell number 0.
If $\mathbb{A}$ is a Turing machine with one-sided tapes, and $x = x_1 \cdots x_n$ an input for $\mathbb{A}$, then the **start configuration of $\mathbb{A}$ on** x is the configuration where all heads are on cell number 0, the imprinting of the first tape reads $\S x_1 \cdots x_n * \cdots$, and the imprinting of any other work tape reads $\S * \cdots$. If $\mathbb{A}$ halts on x, then the **output of $\mathbb{A}$ on** x is defined to be $y_1 \cdots y_m$, where $y_0 y_1 \cdots$ denotes the imprinting of the last work tape in the halting configuration, and m is the biggest natural number such that $y_1, \ldots, y_m$ are all distinct from $*$.
Again, we can view each such Turing machine as computing some partial function $X_0^* \dashrightarrow X_0^*$, where X is the alphabet used. It is then not difficult to show that this notion of computability coincides with the one induced by "two-sided" Turing machines that we developed in detail: To "simulate" a one-sided Turing machine by a two-sided one, just let the two-sided machine at its very first work step print a special symbol on all cells number 0 standing for the § of the one-sided machine and then let it do what the one-sided machine does. At the end, we need to move the last head to the left until we reach cell number 0 and delete the special symbol there (otherwise, it would be counted as part of the output of the two-sided machine), then halt.

Conversely, to simulate a two-sided Turing machine with k tapes by a one-sided one, we take a one-sided Turing machine with $2k$ tapes, two for each work tape of the two-sided machine. The cells with positive indices on the first tape of each such pair of tapes corresponds to the ones with natural number indices of the tape of the two-sided machine, and the cells with positive indices on the second tape to the cells with negative integer indices. During the "simulation", one of the two heads will always be resting on cell number 0 so that the one-sided machine knows whether the corresponding head of the two-sided machine is in the positive or negative part of the tape. This is important, because when the "negative" head is to be moved, the directions "left" and "right" from the two-sided machine must be inversed. In the end, when the two-sided machine would halt, the one-sided simulator still needs to paste the contents of the last two tapes together to give the appropriate output, then halt.
All the other basic notions of recursion theory, such as decidability and recursive enumerability can also be defined analogously to the corresponding notions for two-sided Turing machines and are equivalent to them. We leave the details as an exercise for readers who want to practice recursion theory.

1.5 The Church-Turing Thesis

We have now arrived at the end of this first chapter on recursion theory. Even readers not familiar with Turing machines before will by now at least have developed some "feeling" of what can be done with Turing machines. The purpose of Turing's definition was to give a formalization of the intuitive concept of an algorithm, meaning a process in which some result is "calculated" from a finite amount of data without using any creative power, that is, "mechanically". But has he really succeeded in doing so? Does a notion such as that of Turing computability capture the intuitive notion of computability?
Of course, this is a philosophical question not accessible to formal mathematical treatment, because one of the two notions that are to be compared is not a formal one. Therefore, any formulation of a positive or negative answer to such a question can never be a mathematical theorem, but it is still of some worth. The common belief is that the answer to the above question is to be "yes":

Thesis 1.5.1. *(Church-Turing Thesis)*
Let X be a finite nonempty alphabet with $0, 1 \in X_0$. Then a partial function $f : X_0^* \dashrightarrow X_0^*$ is computable in the intuitive sense if and only if f is (Turing) computable.

Note that this equivalence also implies the equivalence of all other basic recursion-theoretic notions introduced so far with their intuitive counterparts by the standard definitions or characterizations.
If one is not yet convinced of the "truth" of the Church-Turing Thesis, one can proceed to further explore the power of the formal definition by comparing it not just with slight modifications as we did in the last section, but with completely different formalizations. For example, the formal concept of a register machine is closer to the functionality of today's computers, machines closely related to the notion of "algorithmic feasability" (even more so if one idealizes by ignoring any restrictions on time and memory). It can be shown that the notions of Turing-computability and that of computability by a register machine coincide (see, e.g., (16)), which provides evidence that anything which can be computed by a computer can also be computed by a Turing machine.

This will not be part of this thesis, though. We stop at this point and will henceforward feel free to refer to the Church-Turing Thesis in proves of computability, decidability, etc. to make things a bit easier. We will only see the technical details of Turing machines once more when we give a rigorous proof of the "interesting" direction of Matiyasevich's Theorem 2.1.4 (whereas the other direction will be clear by the Church-Turing Thesis).

2 Number-Theoretic Foundations

2.1 Introduction and most elementary properties

In this chapter, based on (11), we study a certain property of sets of tuples of integers:

Definition 2.1.1. *Let $n \in \mathbb{N}, n \geqslant 1$. A subset $S \subseteq \mathbb{Z}^n$ is called **Diophantine** if and only if it is "enumerated by a polynomial", more precisely: There exist $m \in \mathbb{N}$ and a polynomial $P = P(X_1, \ldots, X_n, Y_1, \ldots, Y_m)$ with integer coefficients such that for all tuples $(a_1, \ldots, a_n) \in \mathbb{Z}^n$: $(a_1, \ldots, a_n) \in S \Leftrightarrow \exists b_1, \ldots, b_m \in \mathbb{Z} : P(a_1, \ldots, a_n, b_1, \ldots, b_m) = 0$. The variables $X_1, \ldots, X_n$ are also called the **element parameters**, and $Y_1, \ldots, Y_m$ the **proper variables**. A polynomial whose set of variables is disjointly decomposed into a set of element parameters and a set of proper variables is also called a **Diophantine family**.*

Example 2.1.2. Let $n = 1$. Then the set $S \subseteq \mathbb{Z}$ of even numbers is Diophantine; a possible choice for a Diophantine family witnessing this is $P(X_1, Y_1) = X_1 - 2Y_1$.

We want to establish a connection with Turing machines. For this, first note that we can code integer numbers as strings over $\{0, 1\}$; for example, we can code each natural number n by $[0, \mathrm{bin}(n)]$, and a negative integer of the form $-n$ by $[1, \mathrm{bin}(n)]$; then subsets of $\mathbb{Z}$ become decision problems. However, also $\mathbb{Z}^n$ with $n > 1$ can be coded into $\{0, 1\}^*$, due to the following proposition:

Proposition 2.1.3. *The function* $\mathrm{Cantor} : \mathbb{N}^2 \to \mathbb{N}, (a, b) \mapsto \frac{(a+b)^2+3a+b}{2}$*, is a bijection.*

Proof. Consider the enumeration of type ω of $\mathbb{N}^2$ that lists its elements primarily by the sum of the two components and secondarily by the first entry, in each case in increasing order (i.e., the enumeration goes like this: $(0,0), (1,0), (0,1), (0,2), (1,1), (2,0), \ldots$). To prove the proposition, it suffices to show that $\mathrm{Cantor}(a, b)$ is the rank of (a, b) in this enumeration. But this is easy: First, note that there are $1 + 2 + \cdots + (a + b) = \frac{1}{2}(a + b)(a + b + 1)$ pairs with smaller entry sum that are listed before (a, b). Since the enumeration secondarily goes by the first entry (in increasing order), the rank of (a, b) therefore is $\frac{1}{2}(a + b)(a + b + 1) + a = \frac{(a+b)^2+a+b+2a}{2} = \frac{(a+b)^2+3a+b}{2}$, as we wanted to show. □

This allows us not only to effectively code $\mathbb{N}^2$ into $\mathbb{N}$, but also $\mathbb{N}^n$, by recursively defining functions $\mathrm{Cantor}_n : \mathbb{N}^d \to \mathbb{N}$ by $\mathrm{Cantor}_1 = \mathrm{id}_{\mathbb{N}}$ and $\mathrm{Cantor}_{n+1}(a_1, \ldots, a_{n+1}) = \mathrm{Cantor}_n(a_1, \ldots, a_{n-1}, \mathrm{Cantor}(a_n, a_{n+1}))$. Finally, this allows us to encode $\mathbb{Z}^n$ into $\{0, 1\}^*$ by letting the encoding of $(a_1, \ldots, a_n)$ be $[\epsilon_1, \ldots, \epsilon_n, \mathrm{bin}(\mathrm{Cantor}_n(|a_1|, \ldots, |a_d|))]$, where ϵ_i is either 0 or 1 depending on whether a_i is nonnegative or negative.
Thus it makes sense to ask whether a subset of $\mathbb{Z}^n$ is decidable, recursively enumerable, etc. Our goal for this section is to prove the following astonishing and deep theorem:

Theorem 2.1.4. *(Matiyasevich, 1970)*
A subset $S \subseteq \mathbb{Z}^n$ is Diophantine if and only if S (or, more precisely, its encoding into $\{0, 1\}^*$) is recursively enumerable.

This theorem establishes a link between recursion theory and number theory; it has a number of interesting consequences (like the undecidability of Hilberts's 10th Problem, to which we will come back later) and also proves useful in showing the existence of finitely presented groups with undecidable word problem. Actually, as we will see, it also serves to prove the

so-called Higman Embedding Theorem establishing itself a link between recursion theory and group theory and from which the existence of such groups almost trivially follows.
Note that the implication "⇒" in Theorem 2.1.4 can be proved by specifying an appropriate enumeration machine, the basic idea of which is simple, systematically going through all cases to check in a diagonal fashion; we omit the proof here, but it certainly is a good exercise in recursion theory to at least think of some details of the implementation of this.
However, we are not yet ready to prove the "⇐" from Theorem 2.1.4. Before we can do so, we need to do some of the work that preceded the theorem historically and successively prove more and more powerful properties of the concept of Diophantineness. We start with some elementary observations.
First of all, we will consider a modified notion of Diophantineness, where the coefficients of the polynomial may still be arbitrary integers, but the b_i range only over $\mathbb{N}$. We argue that this notion is equivalent to the one from Definition 2.1.1. First, let $S \subseteq \mathbb{Z}^n$ be Diophantine in the sense of Definition 2.1.1. We need to find a polynomial $Q(X_1, \ldots, X_n, Z_1, \ldots, Z_l)$ such that for all $(a_1, \ldots, a_n) \in \mathbb{Z}^n$:$(a_1, \ldots, a_n) \in S \Leftrightarrow \exists c_1, \ldots, c_l \in \mathbb{N} : Q(a_1, \ldots, a_n, c_1, \ldots, c_l) = 0$. To do so, fix $P(X_1, \ldots, X_n, Y_1, \ldots, Y_m)$ witnessing Diophantineness in the original sense. Then set $l := 2m$ and let $Q(X_1, \ldots, X_n, Z_1, \ldots, Z_{2m}) := P(X_1, \ldots, X_n, Z_1 - Z_2, Z_3 - Z_4, \ldots, Z_{2m-1} - Z_{2m})$. Fix $(a_1, \ldots, a_n) \in \mathbb{Z}^n$. It is clear that if $Q(a_1, \ldots, a_n, Z_1, \ldots, Z_{2m})$ has a root in $\mathbb{N}^{2m}$, then $P(a_1, \ldots, a_n, Y_1, \ldots, Y_m)$ has one in $\mathbb{Z}^m$. But the converse is true as well, since every integer can be expressed as the difference of two appropriate natural numbers. This proves the one direction.
For the other, we need the following theorem from number theory:

Theorem 2.1.5. *Any natural number can be expressed as the sum of four perfect squares (not necessarily all positive).*

Proof. Let M denote the set of all natural numbers that can be expressed as a sum of four perfect squares. By the Euler identity

$$(x_1^2+x_2^2+x_3^2+x_4^2)(y_1^2+y_2^2+y_3^2+y_4^2) = (x_1y_1+x_2y_2+x_3y_3+x_4y_4)^2+(x_1y_2-x_2y_1+x_3y_4-x_4y_3)^2+$$

$$(x_1y_3 - x_3y_1 + x_2y_4 - x_4y_2)^2 + (x_1y_4 - x_4y_1 + x_2y_3 - x_3y_2)^2,$$

it is clear that M is a submonoid of $(\mathbb{N}, \cdot)$ containing 0 and 2. Thus, it suffices to show the statement for odd primes. So fix an odd prime p. We will first show that some multiple $a \cdot p$ of p with $0 < a < p$ is in M. If we knew that $a = 1$, we would be done. So to finish the proof, we will show in a second step that if some multiple $a \cdot p$ of p with $1 < a < p$ is in M, then we can find a smaller factor a', $0 < a' < a$, such that $a' \cdot p$ is in M.
For the first step, note that since $\mathbb{Z}/p\mathbb{Z}$ is a field and thus for each $\alpha \in \mathbb{Z}/p\mathbb{Z}$, the solutions of the quadratic equation $X^2 = \alpha^2$ are precisely α and $-\alpha$, none of the $\frac{p+1}{2}$ numbers $0^2, 1^2, \ldots, (\frac{p-1}{2})^2$ are congruent modulo p. Thus, also none of the $\frac{p+1}{2}$ numbers $-1-0^2, -1-1^2, \ldots, -1-(\frac{p-1}{2})^2$ are congruent modulo p. But together, these are $p+1$ numbers, so some number from the first list must be congruent to some number from the second list, i.e., there are integers $x_1, x_2 \in \{0, \ldots, \frac{p-1}{2}\}$ and a such that $x_1^2 + x_2^2 + 1^2 + 0^2 = x_1^2 + x_2^2 + 1 = a \cdot p$, and it is clear that $0 < a < p$. This concludes the first part of the proof.
For the second step, assume $x_1^2 + x_2^2 + x_3^2 + x_4^2 = a \cdot p$ for some integer a with $1 < a < p$. Let $y_1, \ldots, y_4$ be such that $y_i \equiv x_i (\mathrm{mod}\, a)$ and $-\frac{a}{2} < y_i \leqslant \frac{a}{2}$. Taking the equation modulo a yields $y_1^2 + y_2^2 + y_3^2 + y_4^2 = a' \cdot a$ for some integer a. By choice of the y_i, it is clear that

$a' \leqslant a$, and if $a' = a$, then $y_1 = y_2 = y_3 = y_4 = \frac{a}{2}$. But then there are integers k_i such that $x_i = \frac{2k_i+1}{2}a$, plugging this into the representation of $a \cdot p$ as a sum of four squares and observing that any sum of the squares of four odd numbers is divisible by 4, we get $a \mid p$, a contradiction to $1 < a < p$; thus $a' < a$. Similarly, if $a' = 0$, then $y_1 = y_2 = y_3 = y_4 = 0$ and again, we get that $a \mid p$, a contradiction. So it is now clear that $0 < a' < a$. Now multiplying the two equations and using the Euler identity once more, we get a representation of $a^2a'p$ as a sum of four squares, and by our choice of the y_i, the four squares are all divisible by a^2. Hence, dividing the equation by a^2 yields a representation of $a'p$ as a sum of four squares, as desired. □

Now assume that for some $S \subseteq \mathbb{Z}^n$, there is a polynomial $Q(X_1, \ldots, X_n, Z_1, \ldots, Z_l)$ such that for all $(a_1, \ldots, a_n) \in \mathbb{Z}^n$: $(a_1, \ldots, a_n) \in S \Leftrightarrow \exists c_1, \ldots, c_l \in \mathbb{N} : Q(a_1, \ldots, a_n, c_1, \ldots, c_l) = 0$. We need to specify a polynomial $P(X_1, \ldots, X_n, Y_1, \ldots, Y_m)$ as in Definition 2.1.1. Now by Theorem 2.1.5, for all $(a_1, \ldots, a_n) \in \mathbb{Z}^n$, the polynomial $Q(a_1, \ldots, a_n, Z_1, \ldots, Z_l)$ has a root in $\mathbb{N}^l$ if and only if the system of Diophantine equations, consisting of the equation $Q(a_1, \ldots, a_n, y_1, \ldots, y_l) = 0$ and l additional equations $y_i = z_{i,1}^2 + z_{i,2}^2 + z_{i,3}^2 + z_{i,4}^2$ for $i = 1, \ldots, l$ has a solution in $\mathbb{Z}^{5l}$. But we can convert any parametrized system of Diophantine equations, say with equations $P_i(a_1, \ldots, a_r, w_1, \ldots, w_k) = 0$ for $i = 1, \ldots, s$ (where the parameters and variables specified need not appear in all of the equations), into a single Diophantine equation that is solvable if and only if the system is solvable, namely $\sum_{i=1}^{s} P_i(a_1, \ldots, a_r, w_1, \ldots, w_k)^2 = 0$. This concludes the proof that the two notions of Diophantineness are equivalent.

The method just used also shows that finite intersections of Diophantine sets are Diophantine (note that if the sets of proper variables of the Diophantine families involved are not pairwise disjoint, then we first need to make them), and by forming products, we see that finite unions of Diophantine sets are Diophantine as well. If we view Diophantine sets as defined by some formulas, then this translates as the fact that if we have already verified that two relations are Diophantine, say described by formulas φ and ψ, then the relations described by $\varphi \wedge \psi$ and $\varphi \vee \psi$ also are Diophantine. Furthermore, if $\varphi(x_1, \ldots, x_n)$ describes a Diophantine relation among n integer numbers, then $\exists x_i : \varphi(x_1, \ldots, x_n)$ with $i \in \{1, \ldots, n\}$ describes one among $n - 1$ integers (where x_i becomes a new parameter).

However, complements of Diophantine sets are in general not Diophantine, although it will still take some time before we can show this. For the present, we just need to remember that we cannot use $\neg$ or $\forall$ carelessly in showing that a certain relation is Diophantine using relations we already know to be, but must always argue separately for Diophantineness then.

It also turns out to be fruitful to call a partial function $F : \mathbb{Z}^n \dashrightarrow \mathbb{Z}$ **Diophantine** if and only if the **graph of** F, i.e., the set $\{(x_1, \ldots, x_n, y) \in \mathbb{Z}^{n+1} \mid ((x_1, \ldots, x_n), y) \in F\}$, is Diophantine. Then, in arguing that a certain relation or function is Diophantine, in the formulas that appear we can also use **Diophantine terms**, which are a generalization of variables in the following sense: Any variable and any integer number is a Diophantine term, and when $t_1, \ldots, t_k$ are Diophantine terms and f is a symbol standing for a k-ary Diophantine function, then also $f(t_1, \ldots, t_k)$ is a Diophantine term. Whenever we use an expression of this form, we implicitly assume that the variables are chosen so that the tuple of the values of $t_1, \ldots, t_k$ lies in the domain of f, which is a Diophantine condition by the Diophantineness of f. If one needed to break such a metaargument down to a formal argument that the set in question is Diophantine, one could rewrite the entire Diophantine formula by an appropriate recursion over its complexity into a single Diophantine polynomial witnessing Diophantineness of the

condition; for this, one would have to introduce new variables to represent the values of terms that appear.
In what follows, we will always, unless stated otherwise, assume that the existentially quantified variables of polynomials witnessing Diophantineness range over $\mathbb{N}$. The element parameters will be denoted by X_i, and the proper variables of the polynomials by Y_j. We will now start to build our "library" of Diophantine sets and functions.
Of course, equality is a Diophantine relation, but inequality as well, as witnessed by $(X_1 - X_2)^2 - (Y_1 + 1)$. The relation $\leqslant$ is Diophantine, witnessed by $Y_1 + X_1 - X_2$. Therefore, $<$ is Diophantine as well, since $a < b \Leftrightarrow a \leqslant b \wedge a \neq b$. It now follows immediately that if $R \subseteq \mathbb{Z}^d$ is Diophantine, then so is $R \cap \mathbb{N}^d$.
Divisibility is Diophantine, too, because one can consider $X_1Y_1 - X_2$ (here, one should let the existential quantifiers range over $\mathbb{Z}$, to get divisibility on all of $\mathbb{Z}$, although we will only need it on $\mathbb{N}$). Now for $n, q \in \mathbb{Z}$ with $q \geqslant 1$, let $\mathrm{rem}(n, q)$ denote the remainder of division of n by q. Then since for natural r, we have $r = \mathrm{rem}(n, q) \Leftrightarrow r < q \wedge q \mid n - r$, we get that the function rem is Diophantine. This gives us that nondivisibility is Diophantine, because $d \nmid a \Leftrightarrow (d = 0 \wedge a \neq 0) \vee \mathrm{rem}(a, |d|) > 0$ (and clearly, by what we have already shown, the absolute value is a Diophantine function).
We also get that the integer part function $\mathbb{Z} \times \mathbb{N}\backslash\{0\} \to \mathbb{Z}, (n, q) \mapsto n \operatorname{div} q := \lfloor \frac{n}{q} \rfloor$, is Diophantine, because $b = n \operatorname{div} q \Leftrightarrow bq + \mathrm{rem}(n, q) = n$, and that congruence with respect to a positive modulus is Diophantine: $a \equiv b \pmod m \Leftrightarrow \mathrm{rem}(a, m) = \mathrm{rem}(b, m)$. Finally, the greatest common divisor (and hence also the least common multiple) function is Diophantine: $d = \gcd(a, b) \Leftrightarrow ab \neq 0 \wedge d > 0 \wedge d \mid a \wedge d \mid b \wedge \exists x, y \in \mathbb{Z}[d = ax + by]$.

2.2 Matiyasevich's result about exponentiation

In this section, we will present a modified version of Matiyasevich's original proof of the following theorem:

Theorem 2.2.1. *("Exponentiation is Diophantine")*
The set $\{(a, b, c) \in \mathbb{N}^3 \mid a = b^c\}$ *is Diophantine.*

Obviously, the difficulty here is that there is no bound on the size of the exponent c. This result will be the "key" to Matiyasevich's theorem, and many other nontrivial results eventually leading to its proof will follow rather easily from it. To motivate the proof, we note that already before Matiysevich's breakthrough, earlier works by Julia Robinson had shown that in order to prove Theorem 2.2.1, it would be sufficient to prove the existence of a Diophantine binary relation $R \subseteq \mathbb{N}^2$ of **exponential growth**, i.e., such that for all $a, b \in \mathbb{N} : (a, b) \in R \Rightarrow b < a^a$ and for all $k \in \mathbb{N}$, there exist $a, b \in \mathbb{N}$ with $(a, b) \in R$ such that $b > a^k$. It is not difficult to see that the relation $\{(a, b) \in \mathbb{N}^2 \mid b = \varphi_{2a}\}$, where φ_n denotes the n-th Fibonacci number, is of exponential growth, and Matiyasevich succeeded in showing that it is Diophantine. Following the outline in (11), we will however not consider Fibonacci numbers, but the following second-order recurrent sequences:

Definition 2.2.2. *Let* $b \in \mathbb{N}, b \geqslant 2$. *The* ***special sequence with respect to*** b *is the following recursively defined sequence* $(\alpha_b(n))_{n\in\mathbb{N}}$*:* $\alpha_b(0) = 0, \alpha_b(1) = 1, \alpha_b(n + 2) = b\alpha_b(n + 1) - \alpha_b(n)$.

Note that all these sequences are strictly increasing. They are in some sense "closer" to the first-order recurrent sequences $(\beta_b(n))_{n\in\mathbb{N}}$ defining exponentiation, i.e., with $\beta_b(0) =$

$1, \beta_b(n+1) = b\beta_b(n)$ than the Fibonacci sequence. One could say that the two sequences "asymptotically behave the same", an idea formalized by the following proposition:

Proposition 2.2.3. *(1) Let $b \in \mathbb{N}, b \geqslant 2$. Then for all $n \in \mathbb{N}$: $(b-1)^n \leqslant \alpha_b(n+1) \leqslant b^n$.*
(2) Let $b, c \in \mathbb{N}$. Then $b^c = \lim_{x\to\infty} \frac{\alpha_{bx+4}(c+1)}{\alpha_x(c+1)}$, and $b^c = \alpha_{bx+4}(c+1) \operatorname{div} \alpha_x(c+1)$ for $x = 16(c+1)\alpha_{b+4}(c+1)$.
(3) If $\{(a,b,c) \in \mathbb{N}^3 \mid b \geqslant 4 \wedge a = \alpha_b(c)\}$ is Diophantine, then Theorem 2.2.1 holds.

Proof. For (1): For fixed b, this is an easy induction on n using the recursive definition of $\alpha_b(n)$.
For (2): We will focus on proving the second statement (because only it is interesting from the point of view of Diophantineness), but taking a close look at the bounds we give will also show that the first statement holds. First note that by (1), we have $\frac{\alpha_{bx+4}(c+1)}{\alpha_x(c+1)} \geqslant (\frac{bx+3}{x})^c \geqslant b^c$ for $x \geqslant 2$. Now distinguish the two cases $b = 0$ and $b > 0$.
If $b = 0$ and also $c = 0$, then clearly $\frac{\alpha_{bx+4}(c+1)}{\alpha_x(c+1)} = 1$, which fits together with $0^0 = 1$. If $b = 0$ and $c > 0$, then for $x > 5$: $\frac{\alpha_{bx+4}(c+1)}{\alpha_x(c+1)} \leqslant \frac{4^c}{(x-1)^c} < (\frac{5}{x-1})^c \leqslant 1$, which fits together with $0^c = 0$.
Now consider the case $b > 0$. The subcase $c = 0$ is as for $b = 0$, so suppose $c > 0$. Let $x \geqslant 16c$. Then it is not difficult to show that $\frac{1}{1-\frac{8c}{x}} \leqslant (1 + \frac{16c}{x})$ and that $(1 + \frac{4}{x})^c \leqslant \frac{1}{(1-\frac{4}{x})^c}$, and this gives us the following chain of inequalities: $\frac{\alpha_{bx+4}(c+1)}{\alpha_x(c+1)} \leqslant (\frac{bx+4}{x-1})^c \leqslant (\frac{bx+4b}{x-1})^c = \frac{(1+\frac{4}{x})^c}{(1-\frac{1}{x})^c} b^c \leqslant \frac{b^c}{(1-\frac{1}{x})^c(1-\frac{4}{x})^c} \leqslant \frac{b^c}{(1-\frac{4}{x})^{2c}} \leqslant \frac{b^c}{1-\frac{8c}{x}} \leqslant b^c(1 + \frac{16c}{x})$, where the second-to-last inequality is an application of Bernoulli's inequality. For the statement about the integer quotient to hold, we want the last bound to be smaller than $b^c + 1$, that is, we want $b^c\frac{16c}{x} < 1$, which is equivalent to $x > b^c 16c$. This is certainly satisfied if we choose x as specified, and this choice for x also covers the necessary conditions on x of the other cases.
For (3): This is immediate from the second statement in (2). □

So the main idea behind the proof of Proposition 2.2.3 is to rewrite a statement about asymptotic behavior of the special sequences into a statement about integer quotients, which is of Diophantine character. Our goal thus is to show the "if" in Proposition 2.2.3(3). Before we can do so, we need to better understand the special sequences by proving some results about them. We start by noting that the second-order recurrence in Definition 2.2.2 can be rewritten into a first-order recurrence of matrices:

Lemma 2.2.4. *Let $b \in \mathbb{N}, b \geqslant 2$. For $n \in \mathbb{N}$ define $A_b(n) := \begin{pmatrix} \alpha_b(n+1) & -\alpha_b(n) \\ \alpha_b(n) & -\alpha_b(n-1) \end{pmatrix}$ (where $\alpha_b(-1)$ is understood to be -1, which fits into the recursion) and $\Xi_b := \begin{pmatrix} b & -1 \\ 1 & 0 \end{pmatrix}$. Then $A_b(0) = I_2$ and $A_b(n+1) = A_b(n)\Xi_b$ for $n \in \mathbb{N}$, so that $A_b(n) = \Xi_b^n$. In particular, $\det(A_b(n)) = 1$ so that for all $n \in \mathbb{N}$, the members $\alpha_b(n)$ and $\alpha_b(n+1)$ of the special sequence with respect to b are coprime.*

Proof. This is immediate from Definition 2.2.2. □

Actually, the relation one gets by writing $\det(A_b(n)) = 1$ out explicitly (and replacing $\alpha_b(n+1)$ by $b \cdot \alpha_b(n) - \alpha_b(n-1)$) characterizes successive members of the special sequence with respect to b in the following sense:

Lemma 2.2.5. *Let $b \in \mathbb{N}, b \geqslant 2$. For all $x, y \in \mathbb{N}$, the following are equivalent:*
(1) $x^2 - bxy + y^2 = 1 \wedge y < x$.
(2) $\exists n \in \mathbb{N} : (y = \alpha_b(n) \wedge x = \alpha_b(n+1))$.
In particular, the set $\{(a,b) \in \mathbb{N}^2 \mid b \geqslant 2 \wedge \exists n \in \mathbb{N} : [a = \alpha_b(n)]\}$ is Diophantine.

Proof. "(2)⇒(1)" is immediate from the fact that $\det(A_b(n)) = 1$.
"(1)⇒(2)": By induction on y. If $y = 0$, then $x = 1$, and we can (and must) choose $n = 0$. Now assume the statement has already been shown for all smaller values of y. We expect x, y to be members of the special sequence with respect to b, so we try to "descend down the recursion": Note that $by - y < by - \frac{y^2}{x} < by + \frac{1}{x} - \frac{y^2}{x} = x = by + \frac{1-y^2}{x} \leqslant by$. Let $x_1 := y$ and $y_1 := by - x$ (note that $y_1 \geqslant 0$ because $by \geqslant by + \frac{1-y^2}{x} = x$). Then easy computations show that $x_1^2 - bx_1y_1 + y_1^2 = 1$, so by the induction hypothesis, there is $n_1 \in \mathbb{N}$ such that $x_1 = \alpha_b(n_1+1), y_1 = \alpha_b(n_1)$. But by the way x_1 and y_1 were defined, this means that for $n = n_1+1$, we must have $x = \alpha_b(n+1), y = \alpha_b(n)$, q.e.d. The "in particular" follows immediately, since now we know that this set coincides with $\{(a,b) \in \mathbb{N}^2 \mid b \geqslant 2 \wedge \exists x : [x^2 - abx + a^2 = 1]\}$. □

We need two more arithmetic properties of the special sequences in the form of the following two lemmata:

Lemma 2.2.6. *Let $b_1, b_2, q \in \mathbb{N}$ with $b_1, b_2 \geqslant 2$ and $q \geqslant 1$. Then if $b_1 \equiv b_2 \pmod q$, then also for all $n \in \mathbb{N}$: $\alpha_{b_1}(n) \equiv \alpha_{b_2}(n) \pmod q$. Noting that $\alpha_2(n) = n$ for all $n \in \mathbb{N}$, we have in particular that for all $b \in \mathbb{N}, b \geqslant 3$: $\alpha_b(n) \equiv n \pmod{b-2}$.*

Proof. This is clear by induction on n. □

Lemma 2.2.7. *Let $b, k, m \in \mathbb{N}$ with $b \geqslant 2$. Then: $\alpha_b^2(k) \mid \alpha_b(m) \Rightarrow k \mid m \wedge \alpha_b(k) \mid m$.*

Proof. Let b, k, m accord the assumption. We first show that $k \mid m$, so write $m = kl + n$ with $0 \leqslant n < k$. Then we have:

$$\begin{pmatrix} \alpha_b(m+1) & -\alpha_b(m) \\ \alpha_b(m) & -\alpha_b(m-1) \end{pmatrix} = A_b(m) = \Xi_b^m = \Xi_b^n (\Xi_b^k)^l = A_b(n) A_b^l(k) =$$

$$\begin{pmatrix} \alpha_b(n+1) & -\alpha_b(n) \\ \alpha_b(n) & -\alpha_b(n-1) \end{pmatrix} \begin{pmatrix} \alpha_b(k+1) & -\alpha_b(k) \\ \alpha_b(k) & -\alpha_b(k-1) \end{pmatrix}^l.$$

Considering this identity modulo $\alpha_b(k)$, we get $\alpha_b(m) \equiv \alpha_b(n)\alpha_b^l(k+1) \pmod{\alpha_b(k)}$. Since $\alpha_b(k) \mid \alpha_b(m)$ and $\alpha_b(k), \alpha_b(k+1)$ are coprime, this gives us $\alpha_b(k) \mid \alpha_b(n)$. But $n < k$, so $\alpha_b(n) < \alpha_b(k)$, and this is possible only if $n = 0$, i.e., $k \mid m$.
Now we get $A_b(m) = A_b^l(k) = (\alpha_b(k)\Xi_b - \alpha_b(k-1)I_2)^l = \sum_{i=0}^{l} (-1)^{l-i} \binom{l}{i} \alpha_b^i(k) \alpha_b^{l-i}(k-1) \Xi_b^i$.
Considering the identity modulo $\alpha_b^2(k)$, we get the congruence $\alpha_b(m) \equiv (-1)^{l-1} l \alpha_b(k) \alpha_b^{l-1}(k-1) \pmod{\alpha_b^2(k)}$. Again, since $\alpha_b^2(k) \mid \alpha_b(m)$, we get $\alpha_b^2(k) \mid (-1)^{l-1} l \alpha_b(k) \alpha_b^{l-1}(k-1)$ and thus, dividing both sides by $\alpha_b(k)$ and using that $\alpha_b(k-1), \alpha_b(k)$ are coprime: $\alpha_b(k) \mid l$. But $l \mid m$, so we are done. □

We are now almost ready to prove:

Theorem 2.2.8. *The set $\{(a,b,c) \in \mathbb{N}^3 \mid b \geqslant 4 \wedge a = \alpha_b(c)\}$ is Diophantine.*

One last preparatory remark before we begin the proof: For technical reasons, the proof will not use the usual remainder function rem introduced in the last subsection, but the function arem, giving the so-called **absolutely least remainder**: $\operatorname{arem}(n,q) \equiv \pm n (\mathrm{mod}\ q) \wedge 0 \leqslant \operatorname{arem}(n,q) \leqslant \frac{q}{2}$. It is also a Diophantine function: $r = \operatorname{arem}(n,q) \Leftrightarrow 2r \leqslant q \wedge (q \mid (n-r) \vee q \mid (n+r))$.

Proof of Theorem 2.2.8. We show that a triple $(a,b,c) \in \mathbb{N}^3$ belongs to the set if and only if the following system of Diophantine conditions is satisfiable with appropriate choices of r,s,t,u,v,w,x,y:
(1) $b \geqslant 4$,
(2) $u^2 - but + t^2 = 1$,
(3) $s^2 - bsr + r^2 = 1$,
(4) $r < s$,
(5) $u^2 \mid s$,
(6) $v = bs - 2r$,
(7) $w \equiv b \ (\mathrm{mod}\ v)$,
(8) $w \equiv 2 \ (\mathrm{mod}\ u)$,
(9) $w > 2$,
(10) $x^2 - wxy + y^2 = 1$,
(11) $2a < u$,
(12) $a = \operatorname{arem}(x,v)$,
(13) $c = \operatorname{arem}(x,u)$.
Of course, each of these conditions has a special role. Conditions (2), (3) and (10) serve to introduce in each case two variables as certain successive special sequence values; (4) is an addendum to (3) to additionally control which of the two is the bigger one. Because of this, v introduced by (6) is of a special form that will cause some matrix to reduce in a special way modulo v, see below. Also, note that x and y are values of the special sequence with respect to w (although we need (9) to ensure this), as opposed to u,t,s,r. In view of this and Lemma 2.2.6, conditions (7) and (8) are there to enforce certain helpful congruences between them. Condition (5) will be helpful in view of Lemma 2.2.7, and (12) and (13) (together with (11)) will establish a connection between what was introduced by the other conditions and a and c. This the rough idea for showing that satisfiability of this system implies that the triple (a,b,c) is an element of the specified set. One problem when looking for an appropriate system always is that the conditions, although implying what they should, may be too strong so that one cannot establish the converse implication. But luckily, this is not the case here.
Let us first prove that if (1)-(13) are satisfied, then $a = \alpha_b(c)$. By Lemma 2.2.5, $(1) \wedge (2)$ implies that $u = \alpha_b(k)$ for some k, and $(1) \wedge (3) \wedge (4)$ implies that for some positive m, $s = \alpha_b(m), r = \alpha_b(m-1)$. Now by Lemma 2.2.7 and (5), we get that $u \mid m$. By the recursive definition of $\alpha_b(n)$ and (6), we get $v = b\alpha_b(m) - 2\alpha_b(m-1) = \alpha_b(m+1) - \alpha_b(m-1)$. $(9) \wedge (10)$ gives us $x = \alpha_w(n)$ for some n. Now we can use Lemma 2.2.6 together with $(7) \wedge (8)$ to get $x \equiv \alpha_b(n) \ (\mathrm{mod}\ v)$ as well as $x \equiv n \ (\mathrm{mod}\ u)$. Writing $n = 2lm \pm j$ with $j \leqslant m$ and using Lemma 2.2.4, we have $A_b(n) = (A_b(m)^2)^l A_b(j)^{\pm 1}$. Now by the value of v, $A_b(m) = \begin{pmatrix} \alpha_b(m+1) & -\alpha_b(m) \\ \alpha_b(m) & -\alpha_b(m-1) \end{pmatrix} \equiv -\begin{pmatrix} -\alpha_b(m-1) & \alpha_b(m) \\ -\alpha_b(m) & \alpha_b(m+1) \end{pmatrix} = -(A_b(m))^{-1} \ (\mathrm{mod}\ v)$, where the last equality holds because $\det(A_b(m)) = 1$. This gives us $A_b(n) \equiv \pm A_b(j)^{\pm 1} \ (\mathrm{mod}\ v)$, where all four combinations of the signs are possible. By comparing the left down corners of the matrices, we obtain $\alpha_b(n) \equiv \pm \alpha_b(j) \ (\mathrm{mod}\ v)$. Now $2\alpha_b(j) \leqslant 2\alpha_b(m) \leqslant (b-2)\alpha_b(m) <$

$b\alpha_b(m) - 2\alpha_b(m-1) = v$, so that by (12), $a = \text{arem}(x, v) = \text{arem}(\alpha_b(n), v) = \alpha_b(j)$ (now the role of arem becomes clear; we use it to eliminate the signs). Combining this with (11) gives us $2j \leqslant 2\alpha_b(j) = 2a < u$, so that by (13): $c = \text{arem}(x, u) = \text{arem}(n, u) = j$, where the last equality holds because $n \equiv \pm j \pmod{m}$ and $u \mid m$. Thus $a = \alpha_b(c)$, which is what we wanted to show.
Now we show the converse. So assume a, b, c satisfy $b \geqslant 4 \wedge a = \alpha_b(c)$; we need to show that (2)-(13) are satisfiable with appropriately chosen r, s, t, u, v, w, x, y and let ourselves be guided by the proof of the converse implication.
First, select k large enough so that $\alpha_b(k) > 2a$ and $\alpha_b(k)$ is odd (this is possible because two successive members of the special sequence with respect to b are coprime); set $u := \alpha_b(k)$ so that (11) will be valid. Then set $t := \alpha_b(k+1)$, and by Lemma 2.2.5, (2) will be satisfied. Next let $m := uk$ and set $s := \alpha_b(m), r := \alpha_b(m-1)$. Then (3) and (4) hold.
Also, by the proof of Lemma 2.2.7, $s = \alpha_b(uk) \equiv (-1)^{u-1} u \alpha_b(k) \alpha_b^{u-1}(k-1) \equiv 0 \pmod{u^2}$, where the last congruence holds by choice of u. Therefore, condition (5) is valid as well.
Now let $v := bs - 2r$ so that (6) holds (note that clearly $v \geqslant 0$). In order to take care of (7)-(9), it suffices to show that u and v are coprime (since then we can apply the Chinese Remainder Theorem to get w accordingly). So let $d \mid u$ and $d \mid v$. Then by (5), $d \mid s$, and then by (6), $d \mid 2r$. But by choice of u, d is odd, so $d \mid r$. By (3), s and r are coprime, whence $d = 1$, so u and v are indeed coprime.
For (10), just let $x := \alpha_w(c), y := \alpha_w(c+1)$. Now all that remains are conditions (12) and (13). By (7) and Lemma 2.2.6, $x = \alpha_w(c) \equiv \alpha_b(c) = a \pmod{v}$. But also, $v = bs - 2r > 2\alpha_b(m) \geqslant 2m = 2uk \geqslant 2u > 4a \geqslant 2a$. Therefore, (12) is valid. Finally, by definition of x and Lemma 2.2.6 again, $x \equiv c \pmod{w-2}$, so by (8) $x \equiv c \pmod{u}$, and $2c \leqslant 2\alpha_b(c) = 2a < u$ by (11), whence (13) holds as well, and the proof is complete. □

Proof of Theorem 2.2.1. Clear by Proposition 2.2.3(3) and Theorem 2.2.8. □

2.3 Coding, the factorial function and primes

Recall the Cantor coding function from the first section of this chapter. The fact that it is a polynomial with "almost integer" coefficients makes the components of the inverse function Diophantine; more precisely, for fixed $n, m \in \mathbb{N}\backslash\{0\}$, the function $\text{Elem}_{n,m}$ sending $c \in \mathbb{N}$ to the m-th element of the n-tuple whose Cantor number is c is Diophantine, since for all $a, c \in \mathbb{N}$: $a = \text{Elem}_{n,m}(c) \Leftrightarrow \exists x_1, \ldots, x_{m-1}, x_{m+1}, \ldots, x_n \in \mathbb{N}[2^{2^n}\text{Cantor}_n(x_1, \ldots, x_{m-1}, a, x_{m+1}, \ldots, x_n) = 2^{2^n} c]$, where the factor 2^{2^n} is used to make the coefficients integer.
However, it is not clear why $\text{Elem}_{n,m}(c)$ is Diophantine in all three arguments n, m, c. We need a method to deal with tuples of indefinite length. One example of this would be Gödel coding, which works with the Chinese remainder theorem to code the entries of a tuple of natural numbers as remainders of the division of one number a by several modules b_i that can be chosen so that b_i only depends on one number b and the index i (for the details, see (11), pp. 42f.). Then for Gödel coding, it is not difficult to show that the corresponding element function, here denoted by GElem, is Diophantine in all its arguments.
Gödel coding has, however, another disadvantage: It is difficult to show that the codings of natural operations on tuples (such as concatenation) by corresponding operations on Gödel codes are Diophantine. Thus, the coding of our choice will still be something different, namely positional coding, i.e., coding with cipher representations of numbers.

Our codes will be triples; applying Cantor$_3$ to them would yield numbers as codes. Let a tuple $(a_1, \ldots, a_n) \in \mathbb{N}^n$ be given. For each $b \in \mathbb{N}$ such that $b > a_i$ for all i (so we can view the a_i as ciphers with respect to the base b), the first idea is to code $(a_1, \ldots, a_n)$ by the number $a = \sum_{k=1}^{n} a_k b^{k-1}$. This, however, would not be injective because some a_i at the end can be 0. So the complete coding information is given by the number a (called the **cipher** of the code), the **base** b and the **length** c, put together in a triple (a, b, c). Now not all triples are such positional codes, but the property of being one is Diophantine: For all $a, b, c \in \mathbb{N}$, we have: $\mathrm{Code}(a, b, c) \Leftrightarrow b \geqslant 2 \wedge a < b^c$.
First, we note that also for positional coding, the function Elem mapping a triple (a, b, d) to the d-th element of any tuple coded by (a, b, c) for some appropriate c is Diophantine: $e = \mathrm{Elem}(a, b, d) \Leftrightarrow \exists x, y, z[d = z + 1 \wedge a = xb^d + eb^z + y \wedge e < b \wedge y < b^z]$ (note that to see that this is Diophantine, we need Theorem 2.2.1).
Before we show, as promised, Diophantineness of the codings of "natural" operations, we need the following lemma:

Lemma 2.3.1. *(1) The functions* $\mathbb{N}^2 \to \mathbb{N}, (n, m) \mapsto \binom{n}{m}$, *and* $\mathbb{N} \to \mathbb{N}, n \mapsto n!$, *are Diophantine.*
(2) The set $\mathbb{P}$ *of prime numbers is Diophantine.*

Proof. For (1): Note that by the binomial theorem, $((b+1)^n, b, n+1)$ is a code of the tuple $(\binom{n}{0}, \binom{n}{1}, \ldots, \binom{n}{n})$ for all $b \in \mathbb{N}$ such that $\binom{n}{k} < b$ for all $k \in \{0, \ldots, n\}$; since $\sum_{k=0}^{n} \binom{n}{k} = 2^n$, this will certainly be true for $b = 2^n + 1$. Hence, for all $c, n, m \in \mathbb{N}$, we have: $c = \binom{n}{m} \Leftrightarrow c = \mathrm{Elem}((2^n + 2)^n, 2^n + 1, m + 1)$ so that the first function is Diophantine.
To see that the factorial function is Diophantine, we use the identity $\binom{n}{m} = \frac{n!}{m!(n-m)!}$ to get:

$$m! = \frac{n!}{\binom{n}{m}(n-m)!} = \frac{n(n-1)\cdots(n-m+1)}{\binom{n}{m}} = \frac{n^m}{\binom{n}{m}}(1 - \frac{1}{n}) \cdots (1 - \frac{m-1}{n}).$$

Now view m as fixed and divide both sides by $(1 - \frac{1}{n}) \cdots (1 - \frac{m-1}{n})$. Then the LHS obviously converges to $m!$ for $n \to \infty$, and so does the RHS. We just showed: $\lim_{n\to\infty} \frac{n^m}{\binom{n}{m}} = m!$. Since for large enough n all of the numbers $(1 - \frac{k}{n})$ for $k = 1, \ldots, m-1$ are positive, but smaller than 1, this convergence is from above, so that as in the proof of Proposition 2.2.3, we get $m! = n^m \operatorname{div} \binom{n}{m}$ for "large enough" n; we show that $n = (2m)^{m+2}$ does the job for large enough m, then we are done. First, note that $\frac{1}{1-\frac{k}{n}} = 1 + \frac{k}{n-k}$ for all $k = 1, \ldots, m-1$ as long as $m - 1 < n$, which is certainly true for this choice of n. Thus our goal is to show that $m!(1 + \frac{1}{n-1})(1 + \frac{2}{n-2}) \cdots (1 + \frac{m-1}{n-m+1}) < m! + 1$ for this n. Note that the factors after $m!$ are increasing, so that we can bound this number from above by $m!(1 + \frac{m-1}{n-m+1})^{m-1}$, and we want to show that this is smaller than $m! + 1$. Note that $m!(1 + \frac{m-1}{n-m+1})^{m-1} = m!(1 + \sum_{k=1}^{m-1} \binom{m-1}{k} (\frac{m-1}{n-m+1})^k) \leqslant m!(1 + (m-1)2^{m-1}\frac{m-1}{n-m+1})$, and this will be smaller than $m! + 1$ if and only if $m!2^{m-1}\frac{(m-1)^2}{n-m+1} < 1$, that is, $m! < \frac{n-m+1}{2^{m-1}(m-1)^2} = \frac{(2m)^{m+2}-(m-1)}{2^{m-1}(m-1)^2}$. But apparently, at least for big enough m the RHS is strictly bigger than $\frac{(2m)^{m+1}}{2^{m-1}(m-1)^2} = 4 \cdot \frac{m^{m+1}}{(m-1)^2} > 4 \cdot m^{m-1}$, and it is clear that this is bigger than $m!$.
For (2) This follows from (1) by noting that for all $n \in \mathbb{N}$, n being prime is equivalent to $\gcd(n, (n-1)!) = 1$. □

As a side remark, we now get the existence of a polynomial with integer coefficients whose nonnegative values on nonnegative integer arguments are precisely the prime numbers, by applying this proposition to $S = \mathbb{P}$:

Proposition 2.3.2. *Let $S \subseteq \mathbb{N}$ be Diophantine. Then there is a polynomial $P(X_0, \ldots, X_m)$ with integer coefficients such that for all $(x_0, \ldots, x_m) \in \mathbb{N}^{m+1}$: If $P(x_0, \ldots, x_m) \geqslant 0$, then $P(x_0, \ldots, x_m) \in S$, and all elements of S have a preimage in $\mathbb{N}^{m+1}$ under P.*

Proof. Let $D(X_1, Y_1, \ldots, Y_m)$ be a polynomial witnessing (modified) Diophantineness of S. We define $P(X_0, \ldots, X_m) := (X_0 + 1)(1 - D^2(X_0, \ldots, X_m)) - 1$. First, suppose that for some $(x_0, \ldots, x_m) \in \mathbb{N}$, the value $a := P(x_0, \ldots, x_m) \geqslant 0$. This means $a + 1 = (x_0 + 1)(1 - D^2(x_0, \ldots, x_m))$, so that $D(x_0, \ldots, x_m) = 0$ and $a = x_0$ follows, whence $D(a, x_1, \ldots, x_m) = 0$ and $a \in S$. Conversely, if $a \in S$, then there exist $x_1, \ldots, x_m \in \mathbb{N}$ such that $D(a, x_1, \ldots, x_m) = 0$. Setting $x_0 := a$, we see that $P(x_0, \ldots, x_m) = a$. □

Diophantineness of $\mathbb{P}$ is one of the facts that we need to show Diophantineness of certain operations on codes; the other is the following theorem from elementary number theory:

Theorem 2.3.3. *(Kummer, 1852)*
*Let p be a prime number, ν_p the **p-adic valuation** on $\mathbb{Q}$ (so for $a \in \mathbb{Z}$, $\nu_p(a)$ is the exponent of p in the decomposition of a into prime power factors). Then for all $n, m \in \mathbb{N}$, $\nu_p(\binom{m+n}{m})$ is equal to the number of carries that occur when adding m and n in their p-ary cipher representations.*

Proof. We have that $\nu_p(\binom{m+n}{m}) = \nu_p((m+n)!) - \nu_p(m!) - \nu_p(n!)$. Now it is not difficult to see that for any $a \in \mathbb{N}$, the following holds: $\nu_p(a!) = \sum_{k \geqslant 1} a \operatorname{div} p^k$. This gives us $\nu_p(\binom{m+n}{m}) = \sum_{k \geqslant 1} ((m+n) \operatorname{div} p^k - m \operatorname{div} p^k - n \operatorname{div} p^k)$. Now note that for $a \in \mathbb{N}$, $a \operatorname{div} p^k$ can be obtained by "cutting off" the first k ciphers of the p-ary representation of a. Thus, the k-th summand in the last sum is 0 when there is no carry from the k-th digit (the digit corresponding to the power b^{k-1}) and 1 otherwise. This proves the statement of the theorem. □

Now we are finally ready to prove:

Theorem 2.3.4. *The following relations are all Diophantine:*
(1) $\mathrm{Equal}(a_1, b_1, c_1, a_2, b_2, c_2)$*, which holds if and only if the triples (a_1, b_1, c_1) and (a_2, b_2, c_2) are positional codes of the same tuple.*
(2) $\mathrm{NotGreater}(a_1, b_1, a_2, b_2) :\Leftrightarrow \forall k \in \mathbb{N}\backslash\{0\}[\mathrm{Elem}(a_1, b_1, k) \leqslant \mathrm{Elem}(a_2, b_2, k)]$.
(3) $\mathrm{Small}(a, b, c, e) :\Leftrightarrow \mathrm{Code}(a, b, c) \wedge \forall k \in \mathbb{N}\backslash\{0\}[\mathrm{Elem}(a, b, k) \leqslant e]$.
(4) $\mathrm{Concat}(a, b, c, a_1, b_1, c_1, a_2, b_2, c_2)$*, which holds if and only if the triples $(a, b, c), (a_1, b_1, c_1), (a_2, b_2, c_2)$ are positional codes and the tuple coded by (a, b, c) is the concatenation of the tuples coded by (a_1, b_1, c_1) and (a_2, b_2, c_2).*

Proof. We first consider the following auxiliary relation $\mathrm{PNotGreater}(a_1, a_2, b) :\Leftrightarrow b \in \mathbb{P} \wedge \mathrm{NotGreater}(a_1, b, a_2, b)$. A clever application of Kummer's Theorem 2.3.3 helps us here: If b is a prime, then $\mathrm{NotGreater}(a_1, b, a_2, b)$ holds if and only if there occur no carries when adding $a_2 - a_1$ and a_1 in their p-ary representations, whence $\mathrm{PNotGreater}(a_1, a_2, b) \Leftrightarrow b \in \mathbb{P} \wedge b \nmid \binom{a_2}{a_1}$, so by what we already know, PNotGreater is Diophantine.
We next show that the auxiliary relation $\mathrm{PSmall}(a, b, c, e) :\Leftrightarrow b \in \mathbb{P} \wedge \mathrm{Small}(a, b, c, e)$ is Diophantine. Indeed, this is easy, using Diophantineness of PNotGreater and of the following

partial function $\mathrm{Repeat}(p,q,r)$, defined for all $p,q,r \in \mathbb{N}$ with $p<q$: $\mathrm{Repeat}(p,q,r) := p\frac{q^r-1}{q-1}$. Then: $\mathrm{PSmall}(a,b,c,e) \Leftrightarrow b \in \mathbb{P} \wedge [e \geqslant b \vee \mathrm{PNotGreater}(a,\mathrm{Repeat}(e,b,c),b)]$.
Now suppose we have two bases $b_1 < b_2$, and the positional code (a_2,b_2,c) of some tuple such that all entries of the tuple happen to be even smaller than b_1. So we have $a_2 = \sum_{k=1}^{c} \zeta_k b_2^{k-1}$, and all $\zeta_k \in \{0,\ldots,b_1-1\}$. The tuple $(\zeta_1,\ldots,\zeta_c)$ also has a coding with respect to the base b_1, namely $a_1 = \sum_{k=1}^{c} \zeta_k b_1^{k-1}$, and we see that $a_1 \equiv a_2 \pmod{b_2-b_1}$ ("first condition") as well as $a_1 < b_1^c$ ("second condition"). Thus if b_2 is so large that $b_1^c < b_2 - b_1$, then a_1 is uniquely determined by the two conditions.
Now consider the following auxiliary relation: $\mathrm{Eq}(a_1,b_1,c_1,a_2,b_2,c_2) :\Leftrightarrow \mathrm{Code}(a_1,b_1,c_1) \wedge c_1 = c_2 \wedge \mathrm{PSmall}(a_2,b_2,c_2,b_1-1) \wedge b_1^{c_1} + b_1 < b_2 \wedge a_1 \equiv a_2 \pmod{b_2-b_1}$. Then clearly, Eq is Diophantine, and by the above observations $\mathrm{Eq}(a_1,b_1,c_1,a_2,b_2,c_2) \Rightarrow \mathrm{Equal}(a_1,b_1,c_1,a_2,b_2,c_2)$. The converse is not true because b_2 need not be a prime when $\mathrm{Equal}(a_1,b_1,c_1,a_2,b_2,c_2)$ holds, but we can then "link" the two triples by a third triple where the base is a large enough prime. This gives us $\mathrm{Equal}(a_1,b_1,c_1,a_2,b_2,c_2) \Leftrightarrow \exists x,y,z \in \mathbb{N}[\mathrm{Eq}(a_1,b_1,c_1,x,y,z) \wedge \mathrm{Eq}(a_2,b_2,c_2,x,y,z)]$, so Equal is Diophantine.
The other statements are now easy: $\mathrm{NotGreater}(a_1,b_1,a_2,b_2) \Leftrightarrow \exists x_1,x_2,y,z \in \mathbb{N}[\mathrm{Equal}(a_1,b_1,z,x_1,y,z) \wedge \mathrm{Equal}(a_2,b_2,z,x_2,y,z) \wedge \mathrm{PNotGreater}(x_1,x_2,y)]$ (for "$\Rightarrow$", let y be any prime bigger than both b_1 and b_2, z so big that $a_i < b_i^z$ for $i=1,2$ and x_i the cipher of the positional code with respect to the base y of the tuple coded by (a_i,b_i,z)). Similarly, $\mathrm{Small}(a,b,c,e) \Leftrightarrow \exists x,y \in \mathbb{N}[\mathrm{Equal}(a,b,c,x,y,c) \wedge \mathrm{PSmall}(x,y,c,e)]$. Finally, note that $\mathrm{Concat}(a,b,c,a_1,b_1,c_1,a_2,b_2,c_2) \Leftrightarrow \exists x_1,x_2 \in \mathbb{N}[\mathrm{Equal}(a_1,b_1,c_1,x_1,b,c_1) \wedge \mathrm{Equal}(a_2,b_2,c_2,x_2,b,c_2) \wedge a = x_2 b^{c_1} + x_1 \wedge c = c_1 + c_2]$ (here, the idea is that concatenation on positional codes is easy when they are both with respect to the same base). □

To conclude this subsection about coding, we note that for a fixed base b and any function $F : \{0,1,\ldots,b-1\}^m \to \{0,1,\ldots,b-1\}$, the partial function $F[b](a_1,\ldots,a_m,c)$ defined whenever all (a_i,b,c) are positional codes for $i=1,\ldots,m$ yielding the cipher of the c-tuple $(F(a_{1,1},\ldots,a_{m,1}),\ldots,F(a_{1,c},\ldots,a_{m,c}))$, where a_i is the cipher of the tuple $(a_{i,1},\ldots,a_{i,c})$, is Diophantine.
For this, we introduce b^m "characteristic tuples" $(h_{k_1,\ldots,k_m,1},\ldots,h_{k_1,\ldots,k_m,c})$ for all $k_1,\ldots,k_m \in \{0,1,\ldots,b-1\}$, where $h_{k_1,\ldots,k_m,i}$ is defined to be 1 if $a_{1,i} = k_1 \wedge \ldots \wedge a_{m,i} = k_m$ and 0 otherwise. For $(k_1,\ldots,k_m) \in \{0,1,\ldots,b-1\}^m$ let $h_{k_1,\ldots,k_m}$ be the cipher of the characteristic tuple $(h_{k_1,\ldots,k_m,1},\ldots,h_{k_1,\ldots,k_m,c})$ with respect to the base b. Then the following three identities hold for these ciphers and characterize them among all b^m-tuples of natural numbers:
(1) $a_i = \sum\limits_{k_i \in \{0,\ldots,b-1\}} k_i \cdot \sum\limits_{k_1,\ldots,k_{i-1},k_{i+1},\ldots,k_m \in \{0,\ldots,b-1\}} h_{k_1,\ldots,k_m}$ for $i=1,\ldots,m$.
(2) $\sum\limits_{k_1,\ldots,k_m \in \{0,\ldots,b-1\}} h_{k_1,\ldots,k_m} = \mathrm{Repeat}(1,b,c)$.
(3) $\mathrm{Ortnorm_b}(h_{k_1,\ldots,k_m}, h_{l_1,\ldots,l_m}, c)$ for all distinct tuples $(k_1,\ldots,k_m),(l_1,\ldots,l_m) \in \{0,\ldots,b-1\}^m$,
where $\mathrm{Ortnorm}_b(a_1,a_2,c)$ is the orthonormality relation meaning that (a_1,b,c) and (a_2,b,c) are positional codes of tuples whose entries are 0 or 1 and who never both have 1 at the same position. This relation is Diophantine (we leave it to the reader as an exercise to find a Diophantine representation using results from this subsection). To see the Diophantineness of $F[b](a_1,\ldots,a_m,c)$, just note that the cipher of $(F(a_{1,1},\ldots,a_{m,1}),\ldots,F(a_{1,c},\ldots,a_{m,c}))$ is $\sum\limits_{(k_1,\ldots,k_m) \in \{0,\ldots,b-1\}^m} F(k_1,\ldots,k_m) h_{k_1,\ldots,k_m}$.

2.4 Universal Diophantine equations

Just as there exist universal Turing machines which can simulate any Turing machine, in this section we will show that there are universal Diophantine equations that describe all Diophantine sets of a fixed dimension at once:

Definition 2.4.1. *(1) A **Diophantine family with code parameters** is a polynomial U with integer coefficients whose variables are partitioned into three sets: the **element parameters** $X_1, \ldots, X_n$, the **code parameters** $K_1, \ldots, K_l$ and the **proper variables** $Z_1, \ldots, Z_w$.*
*(2) A Diophantine family with code parameters with notation as in (1) is called a **universal Diophantine family (of n element parameters)** if and only if the following holds: For all Diophantine families $D(X_1, \ldots, X_n, Y_1, \ldots, Y_m)$ with n parameters, there exist natural numbers $k_1^D, \ldots, k_l^D$ such that the Diophantine set described by D coincides with the one described by the specification $U(X_1, \ldots, X_n, k_1^D, \ldots, k_l^D, Z_1, \ldots, Z_w)$, where the X_i are viewed as element parameters and the Z_r as proper variables.*

Before we continue, we note that to show the existence of a universal Diophantine family with n element parameters, it suffices to find one for the Diophantine subsets of $\mathbb{N}^n$ (which is what we will actually do here), i.e., one such that the condition of Definition 2.4.1(2) holds, but where the "Diophantine set described by D" is understood as the intersection of the Diophantine subset of $\mathbb{Z}^n$ described by D in the "usual" sense (actually, already the "modified" version where the proper variables existentially range over $\mathbb{N}$ to make things easier) with $\mathbb{N}^n$. We leave the proof of this as an exercise; the main idea is that one can encode any Diophantine subset of $\mathbb{Z}^n$ by a tuple of 2^n Diophantine subsets of $\mathbb{N}^n$ by fixing the sign in each component and taking absolute values.
The next proposition simplifies the proof of existence a little bit:

Proposition 2.4.2. *(1) Let $n \in \mathbb{N}$. If there exists a universal Diophantine family $U(X_1, \ldots, X_n, K_1, \ldots, K_l, Z_1, \ldots, Z_w)$, then there also exists one with precisely one code parameter.*
(2) If there exists a universal Diophantine family $U(X, K, Z_1, \ldots, Z_w)$ with one element parameter, one code parameter and w proper variables, then for each $n \in \mathbb{N}$, there exists a universal Diophantine family $U_n(X_1, \ldots, X_n, K, Z_1, \ldots, Z_w)$ with n element parameters, one code parameter and also w proper variables.

Proof. For (1): The idea is: Since Cantor coding uses polynomials as coding functions we can use it to code the l code parameters into a single one. Formally, we replace U by $U^2(X_1, \ldots, X_n, K_1, \ldots, K_l, Z_1, \ldots, Z_w) + (K - 2^{2^l}\mathrm{Cantor}_l(K_1, \ldots, K_l))^2$ and now view $K_1, \ldots, K_l$ as proper variables and K as the only code parameter.
For (2): Now we apply Cantor coding to the element parameters to reduce to the case $n = 1$. More precisely, if U is as specified, then set $U_n(X_1, \ldots, X_n, K, Z_1, \ldots, Z_w) := U(2^{2^n}\mathrm{Cantor}(X_1, \ldots, X_n), K, Z_1, \ldots, Z_w)$. We need to argue why U_n is universal in n element parameters. So let $D(X_1, \ldots, X_n, Y_1, \ldots, Y_m)$ be any Diophantine family in n parameters. Now the polynomial $D^2(X_1, \ldots, X_n, Y_1, \ldots, Y_m) + (X - 2^{2^n}\mathrm{Cantor}_n(X_1, \ldots, X_n))^2$ witnesses that the image of the set $\mathcal{D}$ of n-tuples described by D under Cantor coding is a Diophantine subset of $\mathbb{N}$, and so there is a constant $k \in \mathbb{N}$ such that $U(X, k, Z_1, \ldots, Z_w)$ describes this subset. Now for any $a_1, \ldots, a_n \in \mathbb{N}$, by definition $U_n(a_1, \ldots, a_n, k, Z_1, \ldots, Z_w)$ has a root in $\mathbb{N}^w$ if and only if $2^{2^n}\mathrm{Cantor}_n(a_1, \ldots, a_n)$ is the 2^{2^n}-fold of the Cantor number of some element in $\mathcal{D}$, i.e., if and only if $(a_1, \ldots, a_n) \in \mathcal{D}$. □

Thus we can restrict ourselves to the case $n = 1$. The idea is simple: We will code Diophantine families $D(X, Y_1, \ldots, Y_m)$ with one element parameter X by six-tuples (b, c_l, c_r, d, e, f) and potential solutions of the corresponding equations by quintuples (d, e, f, g, h) of natural numbers. Then $U(X, b, c_l, c_r, d, e, f, Z_1, Z_2, Z_3, \ldots, Z_w) = 0$ will say: "The polynomial obtained from the family $D(X, Y_1, \ldots, Y_m)$ coded by (b, c_l, c_r, d, e, f) by plugging in X for the element parameter has the number coded as a potential root by (d, e, f, Z_1, Z_2) as a root.". Universality of U is then clear, but first we must be careful about how we implement these two codings so that everything works.

We first talk about the coding of Diophantine families $D(X, Y_1, \ldots, X_m)$. The numbers d, e, f are easily explained: d is any number larger than the total degree of D, and $e = m$ is the number of proper variables. f is only introduced for notational convenience and is defined via $f := \sum_{k=1}^{e} 2^{d^k}$ (this could be omitted because this defining equality is a Diophantine condition as the reader could show as an exercise; we will not need this fact here though). The triple (d, e, f), which is part of the coding of the Diophantine family, is called the family's **format**.

So far, we have done very well in simplifying the situation by reducing the range of certain quantors from $\mathbb{Z}$ to $\mathbb{N}$, but there is one thing that we do not want to change: The coefficients of the Diophantine families to be considered can and shall be arbitrary integers. Still, we only want to use natural numbers in the coding, so we write $D(X, Y_1, \ldots, Y_m)$ as a difference $C_l(X, Y_1, \ldots, Y_m) - C_r(X, Y_1, \ldots, Y_m)$, where C_l and C_r have coefficients in $\mathbb{N}$; the numbers b as well as c_l and c_r will now be obtained from $C_l(X, Y_1, \ldots, Y_m)$ and $C_r(X, Y_1, \ldots, Y_m)$ respectively through some general method to obtain code numbers b and c from some polynomial $C(X, Y_1, \ldots, Y_m)$ with natural coefficients.

Basically, c is just an encoding for the tuple of coefficients of C via positional coding with respect to an appropriate base b. However, for technical reasons that will reveal themselves later, the entries of the tuple are not just the coefficients of C themselves, but multiplied with certain factors. For the positional coding to work, we demand that $b > d! \cdot \max\{c_{i_0,\ldots,i_m} \mid i_0, \ldots, i_m \in \mathbb{N} \wedge i_0 + \cdots + i_m < d\}$, where $c_{i_0,\ldots,i_m}$ denotes the coefficient of $X^{i_0} Y_1^{i_1} \cdots Y_m^{i_m}$ in C. c is then defined via $c := \sum_{i_0+\cdots+i_m<d} i_0! \cdots i_m!(d - 1 - \cdots - i_m)! c_{i_0,\ldots,i_m} b^{d^{m+1} - i_0 d^0 - \cdots - i_m d^m}$.

If all the numbers fit together as described, the quintuple (b, c, d, e, f) is called a **code** of the polynomial $C(X, Y_1, \ldots, Y_m)$, and if both (b, c_l, d, e, f) and (b, c_r, d, e, f) are codes for polynomials with natural coefficients $C_l(X, Y_1, \ldots, Y_m)$ and $C_r(X, Y_1, \ldots, Y_m)$ respectively such that $C_l - C_r = D$, then (b, c_l, c_r, d, e, f) is called an **extended code** of $D(X, Y_1, \ldots, Y_m)$, and it is clear that every Diophantine family with one element parameter has an extended code.

As for the coding of potential roots of $D(X, Y_1, \ldots, Y_m)$, three of the five numbers used for the code actually depend on the family (not the potential root itself) and have already been defined above, namely d, e and f. For (d, e, f, g, h) to be a code for a potential root, we again use positional coding with g as the base and h as the cipher; for the coding to work, we demand that $g > \max\{1, x_1, \ldots, x_m\}$ and h is then defined to be $\sum_{k=1}^{m} x_k g^{d^k}$.

If we knew that $\mathrm{Format}(d, e, f)$ is Diophantine, we could now also see that the relation $\mathrm{SCode}(d, e, f, g, h)$ holding if and only if (d, e, f) is a format of an appropriate Diophantine family (with one element parameter) and (d, e, f, g, h) is a code for a potential root of the same family (or any family with the same format) is Diophantine, but this is not necessary. Instead, we will define a relation $\mathrm{SCod}(d, e, f, g, h)$ which clearly is Diophantine and coincides with $\mathrm{SCode}(d, e, f, g, h)$ if $\mathrm{Format}(d, e, f)$ holds; this will be sufficient for our purposes.

To see that the following definition of SCod does the job, just observe that if $\mathrm{Format}(d, e, f)$

holds, then the 1s in the 2-ary representation of f with respect to the base 2 are at precisely the same positions where the x_i are supposed to be in the g-ary representation of h if (d, e, f, g, h) is to be a code for a potential root. Now, just set $\mathrm{SCod}(d, e, f, g, h) :\Leftrightarrow \exists t \in \mathbb{N}[\mathrm{Equal}(f, 2, d^e + 1, t, g, d^e + 1) \wedge \mathrm{NotGreater}(h, g, (g-1)t, g)]$.
Using the precise definitions of the two codings to be used, we can now show:

Theorem 2.4.3. *There exists a Diophantine relation* $\mathrm{Solution}(a, b, c_l, c_r, d, e, f, g, h)$ *of nine natural arguments such that whenever* (b, c_l, c_r, d, e, f) *is an extended code of a Diophantine family* $D(X, Y_1, \ldots, Y_m)$ *with one element parameter, the existence of natural numbers* g, h *such that* $\mathrm{Solution}(a, b, c_l, c_r, d, e, f, g, h)$ *holds is equivalent to* $D(a, Y_1, \ldots, Y_m)$ *having a root in* $\mathbb{N}^m$.

Proof. The idea is that we can compute the value of a family $D(X, Y_1, \ldots, Y_m)$ on an argument $(a, y_1, \ldots, y_m)$ just from an extended code (b, c_l, c_r, d, e, f) for D, a code (d, e, f, g, h) for $(y_1, \ldots, y_m)$ as a potential root and the parameter value a without doing any decoding.
For this, let $w > d^{2(e+1)} \cdot a^d \cdot b \cdot d! \cdot h^e$ (note that this condition is Diophantine by what we already know), and first assume that we are dealing with a family $C(X, Y_1, \ldots, Y_m)$ with natural coefficients. Pass from the codes (b, c, d, e, f) of C and (d, e, f, g, h) of $(y_1, \ldots, y_m)$ as a potential root to codes (w, s, d, e, f) and (d, e, f, w, t) of C and $(y_1, \ldots, y_m)$ again, but with respect to the new common base w (note that this transformation of codes is a Diophantine operation, by Diophantineness of Equal).
The value $C(a, y_1, \ldots, y_m)$ is the corresponding linear combination of the monomial values $a^{i_0} y_1^{i_1} \cdots y_m^{i_m}$ with the coefficients of C as scalars. To get a first approximation to this, we consider the number $(1 + aw + t)^{d-1} = (1 + aw^{d^0} + y_1 w^{d^1} + \cdots + y_m w^{d^m})^{d-1} = \sum_{i_0+\cdots+i_m<d} \binom{d-1}{i_0,\ldots,i_m} a^{i_0} y_1^{i_1} \cdots y_m^{i_m} w^{i_0 d^0 + \cdots + i_m d^m}$. Now we already have all the monomial values coded here, but with "wrong" coefficients. To fix this, we multiply this number by $s = \sum_{i_0+\cdots+i_m<d} i_0! \cdots i_m!(d-1-i_0-\cdots-i_m)! c_{i_0,\ldots,i_m} w^{d^{m+1} - i_0 d^0 - \cdots - i_m d^m}$, where $c_{i_0,\ldots,i_m}$ is the coefficient of $X^{i_0} Y_1^{i_1} \cdots Y_m^{i_m}$ in C. Combining terms with the same power of w, we obtain a representation $(1 + aw + t)^{d-1} s = \sum_{k=0}^{2d^{m+1}-1} C_k w^k$.
Now, note two things: First, by the choice of w, it is bigger than any of the C_k (because $\binom{d-1}{i_0,\ldots,i_m} a^{i_0} y_1^{i_1} \cdots y_m^{i_m} \leqslant d! \cdot a^d \cdot h^e$ and $i_0! \cdots i_m!(d-1-i_0-\cdots-i_m)! c_{i_0,\ldots,i_m} \leqslant d! \max\{c_{i_0,\ldots,i_m} \mid i_0 + \cdots + i_m < d\} < b$, and both sums have less than d^{e+1} summands) so that we can now view the C_k as positionally coded with respect to the base w in this product.
Second, it is not difficult to see that

$$C_{d^{e+1}} = \sum_{i_0+\cdots+i_m<d} \binom{d-1}{i_0,\ldots,i_m} i_0! \cdots i_m!(d-1-i_0-\cdots-i_m)! c_{i_0,\ldots,i_m} a^{i_0} y_1^{i_1} \cdots y_m^{i_m} =$$

$$(d-1)! C(a, y_1, \ldots, y_m),$$

and therefore, we can read off the value $C(a, y_1, \ldots, y_m)$ as $\frac{\mathrm{Elem}((1+aw+t)^{d-1} s, w, d^{e+1}+1)}{(d-1)!}$.
Now, supposing we have written $D(X, Y_1, \ldots, Y_m)$ as $C_l(X, Y_1, \ldots, Y_m) - C_r(X, Y_1, \ldots, Y_m)$, note that $D(a, y_1, \ldots, y_m) = 0$ if and only if $C_l(a, y_1, \ldots, y_m) = C_r(a, y_1, \ldots, y_m)$. Therefore, the following definition of Solution does the job: $\mathrm{Solution}(a, b, c_l, c_r, d, e, f, g, h) :\Leftrightarrow \mathrm{SCod}(d, e, f, g, h) \wedge \exists s_l, s_r, t, w \in \mathbb{N}[w > d^{2(e+1)} \cdot a^d \cdot b \cdot d! \cdot h^e \wedge \mathrm{Equal}(c_l, b, d^{e+1}+1, s_l, w, d^{e+1}+1) \wedge \mathrm{Equal}(c_r, b, d^{e+1}+1, s_r, w, d^{e+1}+1) \wedge \mathrm{Equal}(h, g, d^e+1, t, w, d^e+1) \wedge \mathrm{Elem}((1+aw+t)^{d-1} s_l, w, d^{e+1}+1) = \mathrm{Elem}((1+aw+t)^{d-1} s_r, w, d^{e+1}+1)]$. □

Corollary 2.4.4. *There exists $w \in \mathbb{N}$ such that for all $n \in \mathbb{N}$ there exists a universal Diophantine family $U(X_1, \ldots, X_n, K, Z_1, \ldots, Z_w)$ with n element parameters, one code parameter and w proper variables.*

Proof. By Proposition 2.4.2, it suffices to find a universal Diophantine family $U(X, K_1, \ldots, K_l, Z_1, \ldots, Z_w)$ with one element parameter and l code parameters. For this, just set $l := 6$ and let $U(X, K_1, \ldots, K_6, Z_1, Z_2, Z_3, \ldots, Z_w)$ witness Diophantineness of the 9-ary relation Solution, with $X, K_1, \ldots, K_6, Z_1, Z_2$ as element parameters. We view U as a Diophantine family with one element parameter X, 6 code parameters $K_1, \ldots, K_6$ and w proper variables $Z_1, \ldots, Z_w$. Then it is not difficult to see the universality of U by Theorem 2.4.3; the six code parameters corresponding to a given Diophantine family $D(X, Y_1, \ldots, Y_m)$ with one element parameter are, in that order, the entries (b, c_l, c_r, d, e, f) of any extended code of D. □

Note that by our construction, we have not directly constructed the value for the code parameter in an injective way from a Diophantine subset of $\mathbb{N}$, but from a Diophantine family defining this subset. Hence our construction gives rise to a coding not just of Diophantine subsets of $\mathbb{N}$ but even of Diophantine families with one element parameter as natural numbers, by transforming U from the proof of Corollary 2.4.4 into a universal family U_0 with just one code parameter as in the proof of Proposition 2.4.2. Now looking carefully through this proof again, we see that not every natural number is assumed as a code value, but only numbers divisible by 2^{2^6}. But this is not a problem, for if k is not divisible by 2^{2^6}, we simply view it as a code of the family $U_0(X, k, Z_1, \ldots, Z_w)$; then for all $k \in \mathbb{N}$, the family coded by k defines the same Diophantine set as $U_0(X, k, Z_1, \ldots, Z_w)$, which is the only property that we want.
Now this coding in turn gives rise to a coding of parameter-free Diophantine polynomials $P(Y_1, \ldots, Y_m)$, because we can view each such polynomial as an evaluation $D(0, Y_1, \ldots, Y_m)$ of a Diophantine family with one element parameter evaluated at 0 and call $k \in \mathbb{N}$ a **code of** P if and only if k is a code of $D(X, Y_1, \ldots, Y_m)$ as explained in the last paragraph.
Let us now shortly digress on Hilbert's Tenth Problem. We have already talked about coding integers as strings over $\{0, 1\}$, and one can also define a coding of polynomials with integer coefficients as strings over this alphabet. Apart from the fact that the coding should be such that it is decidable whether a string is a code of some polynomial or not, it should have the following property: There exists a Turing machine that rejects inputs which are not binary representations of natural numbers, and on input $\mathrm{bin}(k)$, computes the string code of the parameter-free Diophantine polynomial coded in the above sense by k. It is possible to specify such a string encoding of polynomials; actually, any "reasonable" encoding will do. This property ensures that a set of string codes of parameter-free polynomials is recursively enumerable if and only if the set of string codes of natural number codes of such polynomials is recursively enumerable.
Now Hilbert's Tenth Problem consists of precisely those string codes of parameter-free Diophantine polynomials that have a root. We now show:

Proposition 2.4.5. *(1) The set $\mathcal{H}_0$ of all natural numbers encoding parameter-free Diophantine polynomials with a root is a Diophantine subset of $\mathbb{N}$ whose complement is not Diophantine.*
(2) Matiyasevich's Theorem 2.1.4 implies that Hilbert's Tenth Problem is undecidable.

Proof. (2) is immediate from (1) using the observations above about our string codes for polynomials and referring to 1.2.12.

For (1): Consider the universal Diophantine equation $U_0(X, K, Z_1, \ldots, Z_w)$ with one element parameter described above, and for $k \in \mathbb{N}$ denote by D_k the Diophantine set defined by $U(X, k, Z_1, \ldots, Z_w)$, which coincides with the set defined by the Diophantine family associated to the code k. We first show that the set $\mathcal{H} := \{k \in \mathbb{N} \mid k \notin D_k\}$ is not Diophantine. Assume otherwise. Then we can fix $k_0 \in \mathbb{N}$ such that $\mathcal{H} = D_{k_0}$. But then we get the equivalence $k_0 \in \mathcal{H} \Leftrightarrow k_0 \notin \mathcal{H}$, a contradiction.
To show that the set $\mathcal{H}_0$ is Diophantine, consider the polynomial $U_1(X, Y_1, \ldots, Y_w) := U_0(0, X, Y_1, \ldots, Y_w)$ and note that the Diophantine family $U_1(X, Y_1, \ldots, Y_w)$ with one element parameter X witnesses Diophantineness of $\mathcal{H}_0$.
It remains to show that $\mathbb{N}\backslash\mathcal{H}_0$ is not Diophantine. For this, note that $\mathcal{H}_1 := \mathbb{N}\backslash\mathcal{H} = \{k \in \mathbb{N} \mid k \in D_k\}$ is Diophantine, witnessed by $U(X, X, Y_1, \ldots, Y_w)$. Now we will "reduce $\mathcal{H}_1$ to $\mathcal{H}_0$ in a Diophantine way". Consider any $k \in \mathbb{N}$. In order to decide whether $k \in \mathcal{H}_1$, we can first compute from k a code $\mathcal{K}(k)$ for the parameter-free polynomial $U_0(k, k, Y_1, \ldots, Y_w)$; then $k \in \mathcal{H}_1 \Leftrightarrow \mathcal{K}(k) \in \mathcal{H}_0$, or $k \in \mathcal{H} \Leftrightarrow \mathcal{K}(k) \in \mathbb{N}\backslash\mathcal{H}_0$. Now if $\mathcal{K}$ is a Diophantine function of one argument, then assuming $\mathbb{N}\backslash\mathcal{H}_0$ is Diophantine, we would get that $\mathcal{H}$ is Diophantine, a contradiction.
Thus our goal is to find $\mathcal{K}$ as above such that it is Diophantine. For this, we need to go through our construction of universal Diophantine equations once again to see that actually we can let $\mathcal{K}(k)$ be $K(k, k)$ for some Diophantine polynomial $K(X, Y)$ of two variables.
Indeed, consider for any $p, q \in \mathbb{N}$ the specification $W_{p,q}(Y_1, \ldots, Y_w) := U(p, q, Y_1, \ldots, Y_w)$. First, we construct an extended code (b, c_l, c_r, d, e, f) for this polynomial. The values of d, e, f are constants that do not depend on p, q at all. We carry out the representation as a difference of two polynomials with natural coefficients not for the single $W_{p,q}$, but just once for U and then specialize. This will give us values for the coefficients $c_{l,i_0,\ldots,i_m}, c_{r,i_0,\ldots,i_m}$ as polynomials in p, q with natural number coefficients. Then we can express an appropriate value for b (i.e, such that $b > d!\max(\{c_{l,i_0,\ldots,i_m} \mid i_0 + \cdots + i_m < d\} \cup \{c_{r,i_0,\ldots,i_m} \mid i_0 + \cdots + i_m < d\})$) as a polynomial in p, q and then express the ciphers c_l, c_r as polynomials in p, q. Finally, we just apply the polynomial $2^{2^6}\mathrm{Cantor}_6$ to the tuple $(b(p, q), c_l(p, q), c_r(p, q), d, e, f)$ to get $K(p, q)$. □

2.5 Proof of Matiyasevich's theorem

We have now developed enough theory to finally prove (the "interesting" direction of) Matiyasevich's Theorem 2.1.4. First note that throughout this section we will work with one-sided Turing machines as introduced in the second-to-last section of Chapter 1 for some notational convenience (the reason will become clear soon).
Now for coding purposes, it will be convenient to associate natural numbers with the symbols from our alphabet such that $*$ gets associated with 0; we fix the coding by additionally associating the number 1 to the symbol 0, 2 to the symbol 1 and 3 to the marker §; so the alphabet over which we are working is $X = \{*, §, 0, 1\}$. Finally, for another ease of notation, we will not work with the "economic" version of string encoding of numbers introduced as a motivation (or explanation) of Matiyasevich's theorem, but with a version of **unary encoding**; that is, for $n \in \mathbb{N}$, we represent n itself by the string 001^n, and, for $n > 0$, the negative number $-n$ by 101^n. We leave it as an exercise to show that this change of codes gives an equivalent formulation of Theorem 2.1.4.
The biggest part of the work for the proof will be put into three lemmata beforehand.

Lemma 2.5.1. *("Reduction to the natural number case")*
Assume that for all $n \in \mathbb{N}, n \geqslant 1$, and all subsets $S \subseteq \mathbb{N}^n$, if the set of strings consisting of the canonical codes of elements of S is recursively enumerable, then S is Diophantine. Then Theorem 2.1.4 holds.

Proof Sketch. This is basically the same idea at which we hinted for the exercise to reduce the proof of existence of universal Diophantine equations to the natural number case; note that it is easy to combine the 2^n enumeration machines for the (string codes of) subsets of $\mathbb{N}^n$ that stand for the entire set S into an enumeration machine for the (string code of) S. □

So we only need to take care of the natural number case. For the rest of this section, we fix a subset $S \subseteq \mathbb{N}^n$ such that the set $\tilde{S}$ of string codes of elements of S is recursively enumerable. By Proposition 1.2.19, we can fix a Turing machine $\mathbb{A}$, w.l.o.g. single-tape, such that $\mathbb{A}$ halts on some input $x \in \{0,1\}^*$ if and only if $x \in \tilde{S}$. Also w.l.o.g. we can assume that the set of states of $\mathbb{A}$ is $\{1, 2, \ldots, s\}$ for some $s \in \mathbb{N}, s \geqslant 3$.
We will first show that we can view transition from one configuration of $\mathbb{A}$ to the successor as a Diophantine function. For this, we need to code configurations by tuples of natural numbers. Set $\beta := s + 1$; this will be a fixed base for positional coding. We call $(p, t) \in \mathbb{N}^2$ a **configuration code** if and only if $\mathrm{rem}(t, \beta) = 3$ and there exists some $c \in \mathbb{N}$ such that $\mathrm{Small}(t, \beta, c, 3)$ and $p = i \cdot \beta^k$ for some $i \in \{1, \ldots, s\}$ and $k \in \mathbb{N}$, $k < c$, hold.
The intuition behind this is that p (like "position") with respect to the base β encodes a tuple of the form $(0, \ldots, 0, i, 0, \ldots, 0)$ where the position of i in the tuple is also the position of the head on the tape for the configuration to be encoded and i the state of the configuration, whereas t (like "tape") with respect to β encodes a tuple of the numbers associated with the tape contents of the configuration starting from cell number 0 (which necessarily carries the symbol § = 3) such that all symbols that come after the last symbol listed in the tuple are $*$. Note that each configuration of $\mathbb{A}$ has infinitely many codes. We can now formulate and prove:

Lemma 2.5.2. *("Diophantine simulation of a single step of $\mathbb{A}$")*
There exist Diophantine functions $\mathrm{NextP}, \mathrm{NextT}$ *of two arguments such that whenever (p, t) is a configuration code (for the fixed Turing machine $\mathbb{A}$), then* $(\mathrm{NextP}(p, t), \mathrm{NextT}(p, t))$ *is a code for the successor of the configuration coded by (p, t), where we view any halting configuration of $\mathbb{A}$ as its own successor.*

Proof. We first define from $\mathbb{A}$ three Diophantine functions A, D, Q that describe the different aspects of the transition from one configuration of $\mathbb{A}$ to its successor. If $1 \leqslant i \leqslant s$ and $0 \leqslant j \leqslant 3$, then $A(i, j)$ is defined to be the symbol that $\mathbb{A}$ prints when reading the symbol associated to j in state i; else $A(i, j) := j$. $D(i, j)$ is defined similarly, but describing in the "meaningful" case the direction of the motion of the head. Finally, again in the "meaningful" case, $Q(i, j)$ is the state that $\mathbb{A}$ changes into when reading j in state i.
Using our technique of extending Diophantine functions to tuples, it is easy to see the existence of a Diophantine function NextT as desired because we can get a tuple encoding the tape contents of the successor configuration of the one coded by (p, t) by applying the function A to the corresponding entries of the tuples coded by p and t, whence we can define $t' = \mathrm{NextT}(p, t) :\Leftrightarrow \exists w \in \mathbb{N}[t' = A[\beta](p, t, w)]$.
The existence proof for NextP is similar, but a bit more involved because the k-th entry of a tuple encoding the head position and state of the successor configuration of the one coded

by (p,t) does not only depend on the k-th entries of the tuples coded by p and t, but also on their $(k-1)$-th and $(k+1)$-th entries.
For $c \in \{p,t\}$, set $c^l := c\beta, c^r := c \operatorname{div} \beta$. Note that the first entry of any tuple coded by c^l is 0, whereas its k-th entry, for $k \geqslant 2$, is the $(k-1)$-th entry of any sufficiently long tuple coded by c, and similarly the last entry of any tuple coded by c^r, say of length m, is 0, and its k-th entry for $k < m$ is the $(k+1)$-th entry of any sufficiently long tuple coded by c.
Now we can define a Diophantine function DQ of six natural arguments as follows:

$$DQ(i^l,i,i^r,j^l,j,j^r) := \begin{cases} Q(i^l,j^l) & \text{if } i^l > 0, i = i^r = 0 \text{ and } D(i^l,j^l) = 1, \\ Q(i,j) & \text{if } i^l = 0, i > 0, i^r = 0 \text{ and } D(i,j) = 0, \\ Q(i^r,j^r) & \text{if } i^l = i = 0, i^r > 0 \text{ and } D(i^r,j^r) = -1, \\ 0 & \text{otherwise} \end{cases}$$

and the function DQ allows us to finally define NextP by: $p' = \mathrm{NextP}(p,t) :\Leftrightarrow \exists w \in \mathbb{N}[p' = DQ[\beta](p\beta, p, p \operatorname{div} \beta, t\beta, t, t \operatorname{div} \beta)$. □

Now that we have the Diophantine functions NextP and NextT at our disposal, we define three-argument functions AfterP and AfterT in a simultaneous recursion via

$$\mathrm{AfterP}(0,p,t) := p, \mathrm{AfterT}(0,p,t) := t,$$

$$\mathrm{AfterP}(k+1,p,t) := \mathrm{NextP}(\mathrm{AfterP}(k,p,t), \mathrm{AfterT}(k,p,t)),$$

$$\mathrm{AfterT}(k,p,t) := \mathrm{NextT}(\mathrm{AfterP}(k,p,t), \mathrm{AfterT}(k,p,t).$$

It is clear that if (p,t) is a code for a configuration of $\mathbb{A}$, then $(\mathrm{AfterP}(k,p,t), \mathrm{AfterT}(k,p,t))$ is a code for the k-th successor of this configuration. We are almost done with the proof of Matiyasevich's theorem once we have established:

Lemma 2.5.3. *("Diophantine simulation of several steps of $\mathbb{A}$")*
The functions AfterP *and* AfterT *are Diophantine.*

Proof. We consider the following system of Diophantine conditions with element parameters p,t,k and proper variables $l, p_L, t_L, p_M, t_M, p_R, t_R, p', t'$ (which we will motivate in an instant):
(1) $p < \beta^{l-k-1}$,
(2) $t < \beta^{l-k-1}$,
(3) $p' < \beta^{l-1}$,
(4) $t' < \beta^{l-1}$,
(5) $p_R = \mathrm{NextP}(p_L, t_L)$,
(6) $t_R = \mathrm{NextT}(p_L, t_L)$,
(7) $\mathrm{Concat}(p_L, \beta, kl, p, \beta, l, p_M, \beta, (k-1)l)$,
(8) $\mathrm{Concat}(t_L, \beta, kl, t, \beta, l, t_M, \beta, (k-1)l)$,
(9) $\mathrm{Concat}(p_R, \beta, kl, p_M, \beta, (k-1)l, p', \beta, l)$,
(10) $\mathrm{Concat}(t_R, \beta, kl, t_M, \beta, (k-1)l, t', \beta, l)$.
We will show that for all $p,t,k \in \mathbb{N}$, if (p,t) is a configuration code, then this system has a solution in its proper variables, all solutions are uniquely determined by the value of l, and for each solution $p' = \mathrm{AfterP}(p,t,k)$ and $t' = \mathrm{AfterT}(p,t,k)$; from all this, the Diophantineness of AfterP and AfterT immediately follows.
The proof of existence of a solution also serves to motivate the ideas behind the system. For fixed p,t,k, we consider all the intermediate configurations (p_i,t_i) for $i = 0,\ldots,k$, where

$(p_0, t_0) = (p, t)$, $(p_{i+1}, t_{i+1}) = (\mathrm{NextP}(p_i, t_i), \mathrm{NextT}(p_i, t_i))$. Set $p' := p_k, t' := t_k$. Choose l so big that (1) and (2) hold.
Condition (1) means that in the configuration coded by (p, t), at most the first $l - k - 1$ cells of the tape can carry a proper symbol, and (2) that the head is also resting on one of the first $l - k - 1$ cells. After the following k work steps, at most k more cells can be filled, and the head can only move this many steps, which immediately gives (3) and (4) by our choice of p' and t'. Note, however, that this inequality not only holds of p' and t', but of all p_i and t_i, whence we can view l as the length of tuples coding the intermediate configurations.
Now we define p_L (resp. t_L) as the cipher of the code of the concatenation of the tuples of length l coded by the first k ciphers p_i (resp. t_i) with respect to the base β, and analogously with R instead of L, replacing the word "first" by "last". (p_L, t_L) and (p_R, t_R) are to be viewed as codes of "superconfigurations", that is, configurations of a "supermachine" whose one work tape has k parts, separated by cells marked with § and which has a head on each of the parts to carry out k work steps of $\mathbb{A}$ at once.
The nice thing is that our definitions of the functions NextP and NextT also "work correctly" on codes of superconfigurations, giving us (5) and (6). For (7)-(10), we simply define p_M and t_M as the "common part" of the two superconfigurations, i.e., p_M is the cipher for the code with respect to the base β of the tuples (of length l) coded by $p_1, \ldots, p_{k-1}$, and analogously for t_M; then these last relations are clear by definition. Note that clearly, for the solution specified, $p' = \mathrm{AfterP}(k, p, t)$ and $t' = \mathrm{AfterT}(k, p, t)$.
So all that remains is to show uniqueness of solutions given a fixed value of l (and still, assuming that (p, t) is a configuration code). For this, note that by (7) and (8), the first l entries of the tuples coded by (p_L, β, kl) and (t_L, β, kl) are uniquely determined. Now by (5) and (6), we can basically just pass from these tuples to the ones coded by (p_R, β, kl) and (t_R, β, kl); however, note that in order to know the first m entries of any tuple coded by the cipher $\mathrm{NextP}(p_0, t_0)$ for some $p_0, t_0 \in \mathbb{N}$, one needs to know the first $m + 1$ entries of any tuples coded by p_0 and t_0 respectively, so for the moment, we only get that the first $l - 1$ entries of the tuples coded by (p_L, β, kl) and (p_R, β, kl) are uniquely determined. But by (9) and (10), these coincide with the first $l - 1$ entries of the tuples the codes of which are $(p_M, \beta, (k-1)l)$ and $(t_M, \beta, (k-1)l)$ respectively, and by another application of (7) and (8), we get that even the first $2l - 1$ entries of the tuples with codes (p_L, β, kl) and (t_L, β, kl) are uniquely determined.
Inductively continuing this argument, we see that eventually, we can establish that all entries of the tuples with one of the codes $(p_L, \beta, kl), (t_L, \beta, kl), (p_M, \beta, (k-1)l), (t_M, \beta, (k-1)l)$ and all except possibly the last of the entries of the tuples positionally coded by $(p_R, \beta, kl), (t_R, \beta, kl)$, $(p', \beta, l), (t', \beta, l)$ are uniquely determined; but by (3) and (4), this last entry must be 0, whence we are done. □

Proof of Theorem 2.1.4. Let $i_0, i_1 \in \{1, \ldots, s\}$ be the start and halting state of $\mathbb{A}$. Note that the Diophantine condition $\exists k, r \in \mathbb{N}[\mathrm{Elem}(\mathrm{AfterP}(k, p, t), \beta, r) = i_1]$ holds for some configuration code (p, t) if and only if $\mathbb{A}$ eventually halts after reaching the configuration coded by (p, t).
To get a Diophantine representation of S, all that is left to do is to adequately specify values for p and t in terms of the entries of n-tuples of natural numbers (since we are interested in whether $\mathbb{A}$ halts on the start configurations of the corresponding string codes). It is clear that in this case $p = i_0$, and that, setting $a := 2a_1 + \cdots + 2a_n + 4n + 1$ (a is the number of cells needed to print the string code of $(a_1, \ldots, a_n)$, which

in this case is just the concatenation of the string codes of $a_1, \ldots, a_n$, on the work tape): The tuple positionally coded by (t, β, a) is the concatenation of the tuples whose codes are $(3, \beta, 1), (0, \beta, 4), (\text{Repeat}(1, \beta, 2a_1), \beta, 2a_1), \ldots, (0, \beta, 4), (\text{Repeat}(1, \beta, 2a_n), \beta, 2a_n)$ in this order. Note that we can write this down as a Diophantine condition because n is fixed. $\square$

3 Group-Theoretic Foundations

This chapter, just as the preceding two, is devoted to the discussion of basic concepts which we will need later. After having developed a formal notion of "algorithm" and its most important properties as well as a deep theorem relating recursion theory and number theory, we now turn to the algebraic methods and results involved in proving the undecidability of the word problem.

3.1 Free monoids and group presentations

This section is based on (2). Recall from Chapter 1.1 that we can view any set X as an alphabet. We already stated that the set X^* of all words over X together with the binary operation of "concatenation" forms a monoid; it is called the **free monoid over** X. In this section, we will see that it is in some sense the most general monoid generated by X and how we can use it to get the most general groups generated by X under certain relations. Of course, we need to define formally what this should even mean.

Definition 3.1.1. *Let X be an alphabet.*
*(1) A **monoid relator over** X is just a word over X, i.e., an element of X^*.*
*(2) A **monoid relation over** X is an ordered pair of monoid relators over X; when viewing $(v, w) \in (X^*)^2$ as a monoid relation, we shall write "$v = w$" instead of (v, w).*

These concepts correspond to the syntax of the simple formal language which we want to develop to talk about relations in monoids. As for the semantic level, in order to be able to say whether a monoid relation over X holds or not, we first need to specify what the "variables" stand for:

Definition 3.1.2. *Let M be a monoid, X an alphabet.*
*(1) An **assignment for** X **in** M is just a function $\alpha : X \to M$.*
*(2) For any monoid relator $w \in X^*$ and assignment $\alpha : X \to M$, define the **value of** w **under** α, denoted $\mathrm{val}_\alpha(w)$, recursively by: $\mathrm{val}_\alpha(\varnothing) := 1_M$, and else, writing $w \neq \varnothing$ as $w = w'x$, $x \in X$, $\mathrm{val}_\alpha(w) := \mathrm{val}_\alpha(w') \cdot \alpha(x)$.*
*(3) Let $v = w$ be a monoid relation over X, $\alpha : X \to M$ an assignment. We say that M **under** α **satisfies the relation** $v = w$, in symbols $M \models_\alpha v = w$, if and only if $\mathrm{val}_\alpha(v) = \mathrm{val}_\alpha(w)$.*

The following result, known as the universal property of free monoids, is now immediate:

Proposition 3.1.3. *Let M be a monoid, X an alphabet, $\alpha : X \to M$ an assignment. Then there exists exactly one monoid homomorphism $f : X^* \to M$ such that $f \restriction X = \alpha$.*

Proof. It is not difficult to check that val_α is a monoid homomorphism $X^* \to M$ extending α. Uniqueness is clear because the homomorphism is specified on a generating subset. □

We also have the following easy result:

Proposition 3.1.4. *Let M, M' be monoids, $f : M \to M'$ a monoid homomorphism, X an alphabet and $\alpha : X \to M$ an assignment. Then for any monoid relation $v = w$ over X, the following implication holds: $M \models_\alpha v = w \Rightarrow M' \models_{f \circ \alpha} v = w$.*

Proof. Let $v = x_1 \cdots x_n$, $w = y_1 \cdots y_m$. Then $M \models_\alpha v = w$ means $\alpha(x_1) \cdots \alpha(x_n) = \alpha(y_1) \cdots \alpha(y_m)$. Now just apply f to both sides. □

In words, the statement of Proposition 3.1.4 can be roughly rephrased as "Homomorphisms transfer relations."; one can prove in the framework of universal algebra an according statement for any algebraic structure. Thus, the image of any algebraic structure under a homomorphism can only become "more special" in the sense that the elements corresponding to the images of elements of a generating subset of the domain satisfy even more relations.
Together with Proposition 3.1.3, this gives the "generality" of free monoids mentioned at the beginning: Any relation satisfied by the elements of the canonical generating subset X of X^* must be satisfied by any monoid M under any assignment $\alpha : X \to M$. In fact, the relations satisfied by X^* are only the trivial ones of the form $v = v$ for $v \in X^*$. Of course, this property determines X^* up to isomorphism:

Proposition 3.1.5. *Let M be a monoid, $X \subseteq M$ such that for any monoid M' and any assignment $\alpha : X \to M'$ there exists exactly one homomorphism $f : M \to M'$ extending α. Then there exists an isomorphism $X^* \to M$ extending id_X on X.*

Proof. We can view id_X both as an assignment $X \to X^*$ and $X \to M$. Using the universal property of M and X^* respectively, we get homomorphisms $f : X^* \to M$ and $g : M \to X^*$ both extending id_X on X. We want to show that f and g are inverse isomorphisms; for this purpose, we consider the concatenations $f \circ g : M \to M$ and $g \circ f : X^* \to X^*$. The restrictions to X of both of these concatenations is id_X, but again by the universal property of M and X^* respectively, in each case there can only be one homomorphism with this property, and id_M and id_{X^*} respectively is one. Hence, $f \circ g = id_M$ and $g \circ f = id_{X^*}$, and we are done. □

We next want to adapt these ideas to groups. The first thing we need to modify is the formal language, since we also want to be able to talk formally about inversion. However, since for any group G and any $x_1, \ldots, x_n \in G$, we have $(x_1 \cdots x_n)^{-1} = x_n^{-1} \cdots x_1^{-1}$, we do not lose any "expressiveness" by only considering concatenations of symbols and "formal inverses" of symbols:

Definition 3.1.6. *(1) Let X be an alphabet, X^{-1} a **disjoint copy** of X, i.e., a set Y disjoint from X together with a bijection $i : X \to Y$. We call the elements of X^{-1} the **formal inverses** of the elements of X; instead of $i(x)$, we write x^{-1}.*
From now on, we assume that we have given a set X together with a disjoint copy X^{-1} of X, and we view $X \cup X^{-1}$ as an alphabet.
*(2) A **group relator over** X is just a word over $X \cup X^{-1}$.*
*(3) A **group relation over** X is an ordered pair of group relators; again, instead of (v, w), we rather write "$v = w$".*
Now let G be any group.
*(4) An **assignment for X in G** is just a function $\alpha : X \to G$.*
*(5) For any assignment $\alpha : X \to G$ and word $w \in (X \cup X^{-1})^*$, we define the **value of** w **under** α, denote $\mathrm{val}_\alpha(w)$, by $\mathrm{val}_\alpha(\varnothing) := 1_G$, and else, writing $w = w'x^\epsilon$ with $x \in X$ and $\epsilon \in \{1, -1\}$, $\mathrm{val}_\alpha(w) := \mathrm{val}_\alpha(w') \cdot \alpha(x)^\epsilon$.*
*(6) Let $v = w$ be a group relation over X, $\alpha : X \to G$ an assignment. We say that G **under** α **satisfies the relation** $v = w$, in symbols $G \models_\alpha v = w$, if and only if $\mathrm{val}_\alpha(v) = \mathrm{val}_\alpha(w)$.*

Note that for any set X, there exists a disjoint copy of X; one can, for instance, first identify the elements x of X with $\{\langle 0, x\rangle\}$ as we did before and then set $x^{-1} := \{\langle 1, x\rangle\}$ for any $x \in X$. Of course, when viewing $X \cup X^{-1}$ as an alphabet, $x \in X$ would then correspond to $\{\langle 0, \{\langle 0, x\rangle\}\rangle\}$ and x^{-1} to $\{\langle 0, \{\langle 1, x\rangle\}\rangle\}$.

Now the set $(X \cup X^{-1})^*$ basically contains all we need to form the free group over X, but it is not a group itself, since x and x^{-1} are always viewed just as two different unrelated letters, and xx^{-1} is, as a word of length 2, different from the identity element. To change this situation, we need to identify elements of $(X \cup X^{-1})^*$ in a certain manner:

Definition 3.1.7. *(1) Let M be a monoid. A **monoid congruence on** M is an equivalence relation $\equiv$ on M such that for all $a, a', b, b' \in M$: $a \equiv a' \wedge b \equiv b' \Rightarrow ab \equiv a'b'$.*
*(2) Let G be a group. A **group congruence on** G is just a monoid congruence on G.*

Note that group congruences also commute with inversion, since for any $a, b \in G$, whenever $a \equiv b$, it follows that $a^{-1} = a^{-1} \cdot 1_G = a^{-1}bb^{-1} \equiv a^{-1}ab^{-1} = 1_G \cdot b^{-1} = b^{-1}$. Again, one could define a notion of "congruence" for any algebraic structure, but we only need it for these two cases. The reason to study congruences and not just equivalence relations on algebraic structures is the following proposition, which is immediate from Definition 3.1.7:

Proposition 3.1.8. *(1) Let M be a monoid, $\sim$ an equivalence relation on M. Then $\sim$ is a monoid congruence on M if and only if the following binary operation $\cdot$ on the quotient set $M/\sim$ is well-defined: $[m]_\sim \cdot [m']_\sim := [m \cdot m']_\sim$. In this case, the quotient set endowed with this binary operation is again a monoid called the **quotient monoid of** M **by** $\sim$, and the canonical map $\pi : M \to M/\sim, x \mapsto [x]_\sim$, is a monoid homomorphism.*
(2) Like (1), but replacing every occurrence of the word "monoid" by "group". □

Actually, for group congruences, we have the following correspondence theorem, the (easy) proof of which is left as a strongly recommended exercise for readers unfamiliar with the result:

Theorem 3.1.9. *Let G be a group, $\mathcal{C}(G)$ the set of all group congruences on G and $\mathcal{N}(G)$ the set of all normal subgroups of G. Consider the maps $\nu : \mathcal{C}(G) \to \mathcal{N}(G), C \mapsto [1_G]_C$, where $[g]_C$ denotes the equivalence class of $g \in G$ under C, and $\kappa : \mathcal{N}(G) \to \mathcal{C}(G)$, given by $\forall g, g' \in G : \langle g, g' \rangle \in \kappa(N) \Leftrightarrow g^{-1}g' \in N$ for $N \in \mathcal{N}(G)$ (where the brackets denote the Kuratowski pair). Then:*
(1) The maps ν and κ are inverse bijections.
(2) Let $\equiv$ be a group congruence on G, and make the quotient set $G/\equiv$ into a group as in Proposition 3.1.8. Then $(G/\equiv) \cong (G/\nu(\equiv))$ via $[g]_\equiv \mapsto g \cdot \nu(\equiv)$ for $g \in G$. □

So, group congruences are really just another way of describing normal subgroups.
Returning to our original problem of identifying certain elements of $(X \cup X^{-1})^*$, consider the following definition:

Definition 3.1.10. *Let X be an alphabet, $v, w \in (X \cup X^{-1})^*$. v and w are called **congruent**, in symbols $v \equiv w$, if and only if there exists a finite sequence $(v_1, \ldots, v_n)$ of elements of $(X \cup X^{-1})^*$ such that that $v_1 = v$, $v_n = w$ and for all $i \in \{1, \ldots, n-1\}$, v_{i+1} arises from v_i from a so-called **trivial transformation**, i.e., by insertion or deletion of a successive substring of the form xx^{-1} or $x^{-1}x$ for $x \in X$.*

So, if, for example, $X = \{x, y\}$, then $xyxx^{-1}y^{-1} \equiv x$, with $(xyxx^{-1}y^{-1}, xyy^{-1}, x)$ a finite sequence witnessing this congruence. Now it is not difficult to prove the following:

Proposition 3.1.11. *Let X be an alphabet, $\equiv$ as in Definition 3.1.10. Then $\equiv$ is a monoid congruence on $(X \cup X^{-1})^*$, and the corresponding quotient monoid is a group G satisfying*

the following property: For any group H and any map $f : X \to H$, there exists exactly one homomorphism of groups $\overline{f} : G \to H$ such that $\overline{f} \circ \iota = f$, where $\iota : X \to G$ is the restriction of the canonical projection $\pi : (X \cup X^{-1})^* \to G$ to X.

Proof. It is routine to check that $\equiv$ is a congruence on $(X \cup X^{-1})^*$ so that the quotient will at least be a monoid. But, of course, the inverse of $[x_1^{\epsilon_1} \cdots x_n^{\epsilon_n}]_\equiv$ is given by $[x_n^{-\epsilon_n} \cdots x_1^{-\epsilon_1}]_\equiv$. To show the universal property, fix a group H and a map $f : X \to H$. Define a map $g : X \cup X^{-1} \to H$ extending f by setting $g(x^{-1}) := f(x)^{-1}$ for $x \in X$. Then by the universal property of free monoids, g extends to a homomorphism of monoids $\overline{g} : (X \cup X^{-1})^* \to H$. We now want to define a homomorphism $\overline{f} : G \to H$ by setting $\overline{f}([w]_\equiv) := [\overline{g}(w)]_\equiv$. It will be clear that $\overline{f}$ is a homomorphism of groups once we have verified that it is even well-defined. So suppose we have two words $w, w' \in (X \cup X^{-1})^*$ such that $w \equiv w'$. We need to show: $\overline{g}(w) = \overline{g}(w')$. But w' arises from w by a series of trivial transformations (see Definition 3.1.10), so by an easy induction on the length of the series, we get the desired conclusion. Uniqueness of the homomorphism is clear again. □

The group $G = (X \cup X^{-1})^*/\equiv$ is called the **free group over** X, denoted $F(X)$. Just like the free monoids for the class of all monoids, the free groups are the most general groups in the class of all groups, and are determined up to isomorphism by their universal property. Note that by choosing, for any fixed group H, a generating subset $X \subseteq H$, we can write H as a homomorphic image of a free group (namely $F(X)$) by Proposition 3.1.11; so any group is a quotient of a free group. The kernel of the corresponding epimorphism then contains precisely those equivalence classes of words of $(X \cup X^{-1})^*$ whose value in H is 1_H, i.e., the equivalence classes of relators r such that the relation $r = 1$ holds in H (here, we are writing 1 instead of $\varnothing$). Conversely, given an alphabet X and a set R of group relators over X, we can always construct the most general group generated by X under the relations $r = 1$ for $r \in R$ as a quotient of $F(X)$:

Proposition 3.1.12. *Let X be an alphabet, $R \subseteq (X \cup X^{-1})^*$. Set $G := F(X)/\langle\langle \pi(R) \rangle\rangle$, where $\pi : (X \cup X^{-1})^* \to F(X)$ is the canonical projection and $\langle\langle S \rangle\rangle$ for $S \subseteq G$ is the normal subgroup of G generated by S. Also, consider the assignment $\alpha : X \to G$ given as the composition of the canonical epimorphism $F(X) \to G$ with the restriction of π to X. Then $G \models_\alpha r = 1$ for all $r \in R$. Also, G satisfies the following universal property: For any group H and any assignment $f : X \to H$ such that $H \models_f r = 1$ for all $r \in R$, there exists exactly one homomorphism $\overline{f} : G \to H$ such that $\overline{f} \circ \alpha = f$.*

Proof. Let $r \in R$. Then, $G \models_\alpha r = 1$ just means that $\pi(r)$ is mapped to the identity in G by applying the canonical map $F(X) \to G$. But this is true by construction, since G is the quotient of $F(X)$ by the normal subgroup generated by $\pi(R)$ of which $\pi(r)$ is an element.
For the universal property, fix a group H and an assignment f as in the assumptions. Then by the universal property of free groups, we first get a homomorphism $\widetilde{f} : F(X) \to H$ extending f on the canonical image of X in $F(X)$. But the assumption $H \models_f r = 1$ for all $r \in R$ can be translated to $\pi(r) \in \ker(\widetilde{f})$ for all $r \in R$, whence $\langle\langle \pi(R) \rangle\rangle \subseteq \ker(\widetilde{f})$, so that $\widetilde{f}$ factors through G, giving us a homomorphism $\overline{f} : G \to H$ as desired. Again, uniqueness is clear. □

We now define some notation:

Definition 3.1.13. *Let X be an alphabet, $R \subseteq (X \cup X^{-1})^*$. Then $\langle X \mid R \rangle$ is shorthand for $F(X)/\langle\langle \pi(R) \rangle\rangle$ as in Proposition 3.1.12. We speak of the **group presentation** with **generators** x for $x \in X$ and **relators** r for $r \in R$.*

Sometimes, instead of displaying the right half of a group presentation as a set of relators, it is more convenient to write its elements as group relations; since any group relation can be expressed by saying that a certain group relator represents the identity, this gives an equivalent notion of "group presentation".

Example 3.1.14. (1) $\langle X \mid \varnothing \rangle$ is the free group over X.
(2) $\langle x \mid x^n \rangle$ for $n \in \mathbb{N}$ is $C_n = \mathbb{Z}/n\mathbb{Z}$ (note that $F(\{x\}) \cong \mathbb{Z}$).
(3) $\langle x, y \mid x^2, y^3, xyx^{-1}y^{-1} \rangle = \langle x, y \mid x^2 = y^3 = 1, xy = yx \rangle$ is $C_2 \times C_3$.

The following proposition gives an alternative construction of $\langle X \mid R \rangle$; we leave its proof as an exercise:

Proposition 3.1.15. *Let X be an alphabet, $R \subseteq (X \cup X^{-1})^*$. Define a relation $\equiv$ on $(X \cup X^{-1})^*$ by $v \equiv w$ if and only if w arises from v by a finite sequence of so-called **R-admissible transformations**; these are trivial transformations as in Definition 3.1.10 as well as insertions or deletions of substrings of the form r for any $r \in R$. Then $\equiv$ is a monoid congruence on $(X \cup X^{-1})^*$, and the corresponding quotient monoid is a group G satisfying all the relations $r = 1$ for $r \in R$ under the assignment $f : X \to G$ given by the restriction of the canonical map $(X \cup X^{-1})^* \to G$ to X; furthermore, the homomorphism $\overline{f} : \langle X \mid R \rangle \to G$ obtained from f via the universal property of $\langle X \mid R \rangle$ is an isomorphism.* □

3.2 The word problem for countable presentations

We continue with another section based on (2). Suppose we are given a group presentation $\langle X \mid R \rangle$ and a relator $w \in (X \cup X^{-1})^*$ and asked to determine whether w represents the identity in the presentation or not; alternatively, using Proposition 3.1.15, whether w can be transformed to $\varnothing$ by a finite number of R-admissible transformations. Note that this amounts to being able to decide for any two relators $v, w \in (X \cup X^{-1})^*$ whether they represent the same element of the group presented, since this is equivalent to $v^{-1}w$ representing the identity. We can always answer this question in finite time for the exemplary presentations given in Example 3.1.14; for (2) and (3), this is very easy, and for (1), we refer to:

Proposition 3.2.1. *Let X be an alphabet. A word $w \in (X \cup X^{-1})^*$ is called **trivially reduced** if and only if it does not contain a substring of the form xx^{-1} or $x^{-1}x$ for some $x \in X$. A word w is called a **trivially reduced form of** v if and only if w is trivially reduced and $v \equiv w$ in the sense of Definition 3.1.10. Then the following hold:*
(1) Every word $v \in (X \cup X^{-1})^*$ has exactly one trivially reduced form, denoted by $\overline{v}$.
(2) For any $v \in (X \cup X^{-1})^*$: $F(X) \models_\pi v = 1$ if and only if $\overline{v} = \varnothing$, where π is the restriction of the canonical map $(X \cup X^{-1})^* \to F(X)$ to X.

Proof. (2) follows from (1), since $F(X) \models_\pi v = 1$ by construction of $F(X)$ just means that v can be transformed to $\varnothing$ by a finite sequence of trivial transformations.
For (1): First note that existence of trivially reduced forms is easy: If $v \in (X \cup X^{-1})^*$ is not trivially reduced itself, just keep deleting substrings of the form xx^{-1} or $x^{-1}x$ until the word you get is reduced (formally, we would proceed by induction on the length of v).
Now assume w_1 and w_2 are both trivially reduced forms of v. We need to show $w_1 = w_2$. First note that in particular $w_1 \equiv w_2$, so w_2 arises from w_1 by a finite sequence of trivial transformations which we shall call henceforth a **derivation of** w_2 **from** w_1. Now observe that whenever in such a derivation a deletion step directly precedes an insertion step, we can "swap" the two steps; for example, we can replace any part of the derivation of the

form $(u_1xx^{-1}u_2u_3, u_1u_2u_3, u_1u_2yy^{-1}u_3)$ by $(u_1xx^{-1}u_2u_3, u_1xx^{-1}u_2yy^{-1}u_3, u_1u_2yy^{-1}u_3)$ and the resulting finite sequence of words will still be a derivation of w_2 from w_1; the other cases to be considered are treated similarly; should the insertion happen at the same spot as the deletion, just insert the corresponding string right next to the string to be deleted afterward. Repeatedly applying these replacements, we get a derivation of w_2 from w_1 which first only consists of insertions and then only of deletions. So w_1 and w_2 are both trivially reduced forms of the longest word v' appearing in this derivation and obtainable from v' by a finite sequence of deletions only.
Thus it suffices to show that we cannot arrive at two different trivially reduced words from any word v by using only deletions. This is proved by induction on the length of v. If v itself is reduced, this is trivial, also giving the induction base. Now assume the assertion has already been proved for all words of smaller length than v. Let w_1 and w_2 be trivially reduced and obtained from v by applying only deletions. There are two cases:
Case 1: The first deletions "overlap", i.e., v is of the form u_1xyzu_2 (x, y, z being **symbols**, i.e., either letters from X or formal inverses of letters), and w.l.o.g. in the first step of the derivation of w_1 from v, the substring xy is deleted whereas in the first step of the derivation of w_2, yz is deleted. But then for some letter $t \in X$, either $xy = tt^{-1} \wedge yz = t^{-1}t$ or $xy = t^{-1}t \wedge yz = tt^{-1}$; in both cases, the words obtained after the first deletion step in both derivations are the same! We can now conclude using the induction hypothesis.
Case 2: The first deletions do not overlap. Then v is w.l.o.g. of the form $u_1xx^{-1}u_2yy^{-1}u_3$, and w.l.o.g. in the first step of the derivation of w_1, xx^{-1} is deleted whereas in the other derivation, for the first step yy^{-1} is deleted. But then $u_1u_2u_3$ can be obtained from the word after the first deletion step in both derivations (denoted v_1 and v_2 respectively) with one deletion, and so any reduced form of it obtained using only deletions will be a reduced form of both v_1 and v_2 obtained this way. By the induction hypothesis, we get $w_1 = \overline{u_1u_2u_3}$ (since both are reduced forms of v_1 obtained by a sequence of deletions) and $w_2 = \overline{u_1u_2u_3}$. Hence $w_1 = w_2$, as desired. □

From Proposition 3.2.1, it follows that for any group G and any subset $X \subseteq G$, the subgroup of G generated by X is canonically isomorphic (via the homomorphism from $F(X)$ obtained by the universal property) to $F(X)$ if and only if no nontrivial reduced word w over $X \cup X^{-1}$ becomes 1_G when interpreted in G. In this case, we call X a **free basis** for the subgroup generated by it.
Also by this proposition, we can effectively decide whether a given word v over $X \cup X^{-1}$ represents the identity in $F(X)$ or not, by making trivial deletions in v until we reach a trivially reduced word and check if it is empty or not. So, supposing X is countable (otherwise, there are too many words for an encoding), we can encode words over $X \cup X^{-1}$ in some "reasonable" form (see for instance the next definition) as inputs for a Turing machine which then carries out these work steps to decide whether the word represents the identity in $F(X)$ or not. We now make the following general definition:

Definition 3.2.2. *Let X be a countable alphabet, $R \subseteq (X \cup X^{-1})^*$, and fix some well-order $\lhd$ of type $\alpha \leqslant \omega$ on X.*
*(1) Let ψ denote the unique order isomorphism $X \to \alpha$, and define an injection $\varphi_\lhd : (X \cup X^{-1})^* \to \{0,1\}^*$ by $x_1^{\epsilon_1} \cdots x_n^{\epsilon_n} \mapsto [\delta_1 \mathrm{bin}(\psi(x_1)), \ldots, \delta_n \mathrm{bin}(\psi(x_n))]$, where δ_k is defined to be 0 if $\epsilon_k = 1$ and 1 otherwise. $\varphi_\lhd$ is called the **word Gödelization function under** $\lhd$, its values are the **Gödelizations of words under** $\lhd$.*

(2) The ***word problem for the presentation*** $\langle X \mid R\rangle$ ***w.r.t.*** $\lhd$*, denoted* $\mathrm{WP}_{X,R,\lhd}$*, is the set of all word Gödelizations under* $\lhd$ *which represent the identity in* $\langle X \mid R\rangle$*.*

Our observations from above show that $WP_{X,\varnothing,\lhd}$ is decidable for any countable X and any well-order $\lhd$ on X. However, in general, this is not independent of the choice of $\lhd$; for example, take any countably infinite X and fix a well-order $\lhd_0$ on X, giving us an enumeration of its elements as x_n, $n \in \mathbb{N}$. Now define $R := \{x_n^n \mid n \in \mathbb{N}\}$. We shall see later that $\langle X \mid R\rangle$ is a so-called free product, and by the normal form theorem for free products, for any $n, k \in \mathbb{N}$, we have that $x_n^k = 1$ in $\langle X \mid R\rangle$ if and only if $n \mid k$. From this, it is not difficult to see that for any two well-orders $\lhd_1, \lhd_2$ on X with $\lhd_1 \neq \lhd_2$, we have $WP_{X,R,\lhd_1} \neq WP_{X,R,\lhd_2}$, giving us $2^{\aleph_0}$ "different word problems" for this presentation. But there are only $\aleph_0$ many Turing machines overall, so not all $WP_{X,R,\lhd}$ for $\lhd$ a well-order on X can be decidable.
However, we shall only be interested in presentations $\langle X \mid R\rangle$ with $|X| < \aleph_0$, and it is easy to see that for these, the decidability of the word problem is independent of the choice of the well-order $\lhd$ (because for any two well-orders $\lhd_1$ and $\lhd_2$ on X, there is a Turing machine "translating" Gödelizations with respect to $\lhd_1$ into the corresponding Gödelizations with respect to $\lhd_2$). Thus, henceforward we shall only speak of the "decidability of the word problem of the presentation $\langle X \mid R\rangle$".
Actually, we can loosen up our mode of speech in the case $|X| < \aleph_0$ even more, according to the next proposition:

Proposition 3.2.3. *Let* X, Y *be finite alphabets,* $R_1 \subseteq (X \cup X^{-1})^*$*,* $R_2 \subseteq (Y \cup Y^{-1})^*$ *such that* $\langle X \mid R_1\rangle \cong \langle Y \mid R_2\rangle$*. Then the word problem of* $\langle X \mid R_1\rangle$ *is decidable if and only if the word problem of* $\langle Y \mid R_2\rangle$ *is decidable.*

Proof. By symmetry, it suffices to show one direction of this equivalence. So assume that the word problem of $\langle X \mid R_1\rangle$ is decidable. Fix well-orders $\lhd_X$ and $\lhd_Y$ on X and Y respectively. Now fix a Turing machine $\mathbb{A}$ deciding $WP_{X,R,\lhd_X}$, and fix an isomorphism $f : \langle Y \mid R_2\rangle \to \langle X \mid R_1\rangle$. Now, since Y is finite, f is "computable" in the sense that given any word over $Y \cup Y^{-1}$, we can effectively compute a word over $X \cup X^{-1}$ representing the image in $\langle X \mid R_1\rangle$ under f of the element represented by the given word in $\langle Y \mid R_2\rangle$; just "store", for any symbol $t \in Y \cup Y^{-1}$, a word over $X \cup X^{-1}$ representing the image of the element represented by t (this needs finitely many states). The machine $\mathbb{B}$ "computing f" first prints 1 as the very first output bit and then just reads its input symbolwise and concatenates stored values accordingly to form the rest of the output. But then the function really computed by $\mathbb{B}$ (in the sense of Definition 1.2.6) is a reduction of $\mathrm{WP}_{Y,R_2,\lhd_Y}$ to $\mathrm{WP}_{X,R_1,\lhd_X}$. □

By Proposition 3.2.3, we can define (un-)decidability of the word problem as an abstract property of finitely generated groups, by saying that a finitely generated group G has decidable word problem if and only if for some (and thus, for any) finite generating subset X of G and well-order $\lhd$ on X, $\mathrm{WP}_{X,\ker\pi,\lhd}$ is decidable, where π denotes the canonical epimorphism $\pi : (X \cup X^{-1})^* \to G$.
Note that presentations even of finitely generated groups with decidable word problem can have undecidable word problem if they use infinitely many generators and a "pathological" set of relators. For example, fix any recursively enumerable but not recursive subset $S \subseteq \mathbb{N}$ (see Theorem 1.3.12) and note that by the observations from the beginning of this section and the mode of speech just introduced in the last paragraph, the free group $F(x, y)$ over 2 generators has decidable word problem. However, the following is a presentation of this

group which has undecidable word problem (because S can be reduced to it): $\langle x, y, z_n\ (n \in \mathbb{N}) \mid x^{-1}z_n\ (n \in S), y^{-1}z_n\ (n \in \mathbb{N}\backslash S)\rangle$.
However, there are also far less "pathological" presentations with undecidable word problem. Among the most harmless presentations are certainly the finite ones, where a presentation $\langle X \mid R\rangle$ is called **finite** if and only if both X and R are finite; this is even stronger than just demanding X to be finite and R to be recursive (so one cannot "pull a trick" like the one in the last paragraph). And yet, there are finite presentations (and thus, finitely generated groups) with undecidable word problem, but this is far less trivial. Before we can discuss how to prove this, we need some more concepts from combinatorial group theory, introduced in the next few sections.

3.3 HNN extensions

This section (and the next one) is based on (10). HNN stands for "Higman-Neumann-Neumann", the names of the three mathematicians who introduced this notion in 1949. Suppose we are given a group G, two isomorphic subgroups $A, B \leqslant G$ and an isomorphism $\varphi : A \to B$. Then φ is not necessarily the restriction of the conjugation by some element of G to A, but we can construct a bigger group, called the **HNN extension of G w.r.t. A, B and φ** and denoted $G^{*(\varphi)}$, in which this is the case. Actually, its definition is rather self-suggesting:

Definition 3.3.1. *Let G be a group, $A, B \leqslant G$, $\varphi : A \to B$ an isomorphism. Then $G^{*(\varphi)}$ is defined as $\langle G, t \mid t^{-1}at\varphi(a)^{-1}(a \in A)\rangle$. In this construction, G is called the **base** of the HNN extension, t is called the **stable letter** and A, B are the **associated subgroups**.*

Note that the notation used in Definition 3.3.1 is not completely precise; what we mean is that we take any presentation of G, add another generator t and, for any $a \in A$ (or, equivalently, just for the elements a of some generating subset of A), choose words w_a and v_a over the generators of the presentation of G and their formal inverses representing a and $\varphi(a)$ respectively and add the relator $t^{-1}w_a t v_a^{-1}$, where v_a^{-1} is the formal inverse of the word v_a (write down the word in the reverse order and change the sign of all exponents). We leave it as an exercise to show that all presentations obtained in this way are isomorphic.
All the desired properties from the motivation before Definition 3.3.1 are clear by construction except for the fact that G canonically embeds into $G^{*(\varphi)}$. This actually is not so trivial, and we shall show it by means of the so-called normal form theorem for HNN extensions, reducing the syntax of HNN extensions to the syntax of the base group (so from a logical viewpoint, we shall be doing proof theory for the formal system given by HNN presentations). Let us first define the important notions:

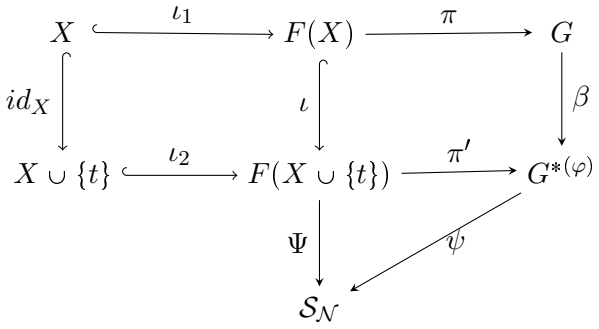
Definition 3.3.2. *Let G be a group, $A, B \leqslant G$, $\varphi : A \to B$ an isomorphism, t a stable letter.*
*(1) A finite sequence $(g_0, t^{\epsilon_1}, g_1, \ldots, t^{\epsilon_n}, g_n)$ with $n \geqslant 0$, $g_i \in G$ and $\epsilon_j \in \{1, -1\}$ is called **HNN reduced** if and only if it does not contain a successive subsequence of the form (t^{-1}, g_i, t) with $g_i \in A$ or (t, g_j, t^{-1}) for $g_j \in B$.*
*(2) Fix a set of representatives $\mathcal{A}$ of the right cosets of A and $\mathcal{B}$ of the right cosets of B in G respectively such that $1_G \in \mathcal{A} \cap \mathcal{B}$. A finite sequence $(g_0, t^{\epsilon_1}, g_1, \ldots, t^{\epsilon_n}, g_n)$ as in (1) is called an **HNN normal form w.r.t. $\mathcal{A}$ and $\mathcal{B}$** if and only if $\epsilon_i = -1 \Rightarrow g_i \in \mathcal{A}$, $\epsilon_i = 1 \Rightarrow g_i \in \mathcal{B}$ and the sequence does not contain a successive subsequence of the form $(t^{\epsilon}, 1, t^{-\epsilon})$.*

The idea behind these two concepts is that any element of the corresponding HNN extension has a reduced representation (this is easy) from which one can read whether the element is the identity in the extension or not. However, reduced representations are not unique since, e.g., any normal form is a reduced sequence (because otherwise, the existence of a subsequence (t^{-1}, a, t) with $a \in A$ together with the first of the two implications from Definition 3.3.2(2) gives $a = 1_G$, a contradiction to the last property demanded, and similarly for the other case), but different choices for $\mathcal{A}$ and $\mathcal{B}$ give different normal forms. However, when fixing $\mathcal{A}$ and $\mathcal{B}$, then normal forms become canonical reduced form representations, with each element of the HNN extension having precisely one representation by a normal form:

Theorem 3.3.3. *(Normal Form Theorem for HNN Extensions)*
Let G be a group, $A, B \leqslant G$, $\varphi : A \to B$ an isomorphism. Then:
(1) G canonically embeds into $G^{*(\varphi)}$. If $(g_0, t^{\epsilon_1}, g_1, \ldots, t^{\epsilon_n}, g_n)$ is some sequence such that the product of its entries is the identity in $G^{*(\varphi)}$ and $n \geqslant 1$, then the sequence is not reduced.
(2) Fix sets $\mathcal{A}$ and $\mathcal{B}$ of right coset representatives as in Definition 3.3.2. Then for every element w of $G^{*(\varphi)}$, there is exactly one normal form w.r.t. $\mathcal{A}$ and $\mathcal{B}$ such that w is the product of the entries of this normal form.

Proof. Let us first make more explicit what (1) and (2) mean in the strictly formal sense. Fix a presentation $G = \langle X \mid R \rangle$, and note that by Proposition 3.2.1, $F(X)$ canonically embeds into $F(X \cup \{t\})$ by the (well-defined) map ι sending $[w]_{\equiv_1} \mapsto [w]_{\equiv_2}$, where $\equiv_1$ is the trivial word congruence on $(X \cup X^{-1})^*$ and $\equiv_2$ the one on $(X \cup X^{-1} \cup \{t, t^{-1}\})^*$. Further note that we have canonical epimorphisms $\pi : F(X) \to G$ and $\pi' : F(X \cup \{t\}) \to G^{*(\varphi)}$. But $\iota(\ker(\pi)) \subseteq \ker(\pi')$, since all the defining relators of the presentation of G also appear in the presentation of $G^{*(\varphi)}$. Thus, we get a canonical homomorphism $\beta : G \to G^{*(\varphi)}$ sending $\pi(w) \mapsto \pi'(\iota(w))$, and the first claim in (1) is that β is an embedding. So the g_i in the sequences to be discussed literally are elements of G, and when we compute which element in $G^{*(\varphi)}$ is denoted by the sequence, we use their images under β (together with the images of the symbols t and t^{-1} under π').
We first prove that (2) implies (1), so we will only have to show (2) afterward. First, since for any $g \in G$, the 1-tuple (g) is a normal form, it is clear that β needs to be injective if (2) holds. For the second statement, note that we can tranform every reduced form into a normal form w.r.t. $\mathcal{A}$ and $\mathcal{B}$ representing the same element in the HNN extension by working "from the right to the left"; if $n = 0$, there is nothing to show, and otherwise, first look at g_n. If $\epsilon_n = -1$, then we want to replace g_n by $g_n' \in \mathcal{A}$; so write $g_n = a_n g_n'$, where $a_n \in A$ and $g_n' \in \mathcal{A}$. The defining relators $t^{-1}at\varphi(a)^{-1}$, $a \in A$, give the relations $t^{-1}a = \varphi(a)t^{-1}$ for $a \in A$, so viewing this as "quasi-commuting relations", we can "swap" a_n with the preceding t^{-1} by turning it into $\varphi(a_n)$. The case $\epsilon_n = 1$ is treated analogously. Now the end of our reduced form already "looks like" a normal form, and continuing inductively, we can turn the entire reduced form into a normal form with the same number of symbols t, t^{-1} and even the same sequence of exponents ϵ_i; note that the last property in the definition of "normal form" will also never be violated in the sequence we construct as long as we start with a reduced sequence (e.g., in a subsequence (t^{-1}, g_i, t), by reducedness g_i will never be in A, but since the symbol afterward is t, what we bring together with g_i by an application of a quasi-commuting relation is an element of A, so the representative from $\mathcal{A}$ that will take the place of g_i in the normal form that we construct is different from 1_G; the other case is treated analogously). Now, assuming we have a sequence like in the assumption of the second part of (1) and it is reduced, then

applying this procedure gives us a normal form of the same length which represents $1_{G^{*(\varphi)}}$. But (1_G) is also a normal form representing this element, contradicting our assumption (2).
Now we show (2), i.e., uniqueness of normal form representations. We do this using a standard argument introduced by Artin and van der Waerden, letting the group for which we want to show a normal form theorem act on the set of normal forms by "simulation of left multiplication". So, let $\mathcal{N} = \mathcal{N}_{\mathcal{A},\mathcal{B}}$ be the set of all normal forms w.r.t. $\mathcal{A}$ and $\mathcal{B}$. We now define a homomorphism $\psi : G^{*(\varphi)} \to \mathcal{S}_{\mathcal{N}}$. Since $G^{*(\varphi)}$ is a quotient of $F(X \cup \{t\})$, we shall first define a homomorphism $\Psi : F(X \cup \{t\}) \to \mathcal{S}_{\mathcal{N}}$ and gain ψ by factoring Ψ through $G^{*(\varphi)}$. By the universal property of free groups, in order to define Ψ, just define it on the generating set $X \cup \{t\}$, and it will be no problem to extend this mapping information to a homomorphism on the free group. The situation we shall have in the end is depicted by the following commutative diagram:



For $x \in X$, define $\Psi(\iota_2(x))$ by $\Psi(\iota_2(x))(g_0, t^{\epsilon_1}, \ldots, t^{\epsilon_n}, g_n) := (\pi(x)g_0, t^{\epsilon_1}, \ldots, t^{\epsilon_n}, g_n)$; it is clear that $\Psi(\iota_2(x)) \in \mathcal{S}_{\mathcal{N}}$, with the inverse permutation given by $(g_0, t^{\epsilon_1}, \ldots, t^{\epsilon_n}, g_n) \mapsto (\pi(x)^{-1}g_0, t^{\epsilon_1}, \ldots, t^{\epsilon_n}, g_n)$.
The definition of $\Psi(\iota_2(t))$ is a bit more involved. Let $(g_0, t^{\epsilon_1}, \ldots, t^{\epsilon_n}, g_n)$ be a normal form. If $\epsilon_1 = -1$ and $g_0 \in B$, then define $\Psi(\iota_2(t))(g_0, t^{\epsilon_1}, \ldots, t^{\epsilon_n}, g_n) := (\varphi^{-1}(g_0)g_1, t^{\epsilon_2}, g_2, \ldots, t^{\epsilon_n}, g_n)$, thus "making the necessary reduction". Otherwise write $g_0 = bg_0'$ with $b \in B$ and $g_0' \in \mathcal{B}$, and define $\Psi(\iota_2(t))(g_0, t^{\epsilon_1}, \ldots, t^{\epsilon_n}, g_n) := (\varphi^{-1}(b), t, g_0', t^{\epsilon_1}, g_1, \ldots, t^{\epsilon_n}, g_n)$.
First, we should check that $\Psi(\iota_2(t))$ is a permutation of $\mathcal{N}$. To do this, we explicitly write down the inverse permutation called $\Psi'(\iota_2(t))$ and check that it is inverse to $\Psi(\iota_2(t))$. The definition of $\Psi'(\iota_2(t))$ is dual to the one of $\Psi(\iota_2(t))$; let $(g_0, t^{\epsilon_1}, \ldots, t^{\epsilon_n}, g_n)$ be a normal form. If $\epsilon_1 = 1$ and $g_0 \in A$, then "make the necessary reduction" setting $\Psi'(\iota_2(t))(g_0, t^{\epsilon_1}, \ldots, t^{\epsilon_n}, g_n) := (\varphi(g_0)g_1, t^{\epsilon_2}, g_2, \ldots, t^{\epsilon_n}, g_n)$. Otherwise, write $g_0 = ag_0'$ with $a \in A$ and $g_0' \in \mathcal{A}$ and set $\Psi'(\iota_2(t))(g_0, t^{\epsilon_1}, \ldots, t^{\epsilon_n}, g_n) := (\varphi(a), t^{-1}, g_0', t^{\epsilon_1}, \ldots, t^{\epsilon_n}, g_n)$.
Now we need to verify that $\Psi'(\iota_2(t)) \circ \Psi(\iota_2(t)) = \Psi(\iota_2(t)) \circ \Psi'(\iota_2(t)) = id_{\mathcal{N}}$. However, we shall only show $\Psi'(\iota_2(t)) \circ \Psi(\iota_2(t)) = id_{\mathcal{N}}$, for the other statement can be treated completely analogously. So, let $(g_0, t^{\epsilon_1}, \ldots, t^{\epsilon_n}, g_n) \in \mathcal{N}$. There are two cases to be considered. First assume that the first case in the definition of $\Psi(\iota_2(t))$ applies. So $g_0 \in B$ and $\epsilon_1 = -1$, and an application of $\Psi(\iota_2(t))$ transforms the normal form into $(\varphi^{-1}(g_0)g_1, t^{\epsilon_2}, g_2, \ldots, t^{\epsilon_n}, g_n)$. Now we cannot have $\epsilon_2 = 1$ and $g_1 \in A$ simultaneously (since then the original normal form would have contained a subsequence of the form $(t^{-1}, 1, t)$), so the second case in the definition of $\Psi'(\iota_2(t))$ applies, and after applying $\Psi'(\iota_2(t))$, we get (noting that $g_1 \in \mathcal{A}$) $(\varphi(\varphi^{-1}(g_0)), t^{-1}, g_1, \ldots, t^{\epsilon_n}, g_n)$, i.e., the original normal form. In the other case, an application of $\Psi(\iota_2(t))$ to the normal form gives $(\varphi^{-1}(b), t, g_0', t^{\epsilon_1}, \ldots, t^{\epsilon_n}, g_n)$, where $g_0 = bg_0'$ with $b \in B$ and $g_0' \in \mathcal{B}$. Now it is clear that the first case in the definition of $\Psi'(\iota_2(t))$ applies to this normal form, so after applying $\Psi'(\iota_2(t))$, we get $(\varphi(\varphi^{-1}(b))g_0', t^{\epsilon_1}, g_1, \ldots, t^{\epsilon_n}, g_n)$, which

again is just the original normal form.
Now extend this mapping information to a homomorphism $\Psi : F(X \cup \{t\}) \to \mathcal{S}_{\mathcal{N}}$. In order to show that Ψ factors through $G^{*(\varphi)}$, one first shows that for all $w \in F(X)$ with $\pi(w) = 1_G$, $\Psi(\iota(w)) = id_{\mathcal{N}}$, which is clear by the definition of Ψ, and then, using similar case distinctions as above, that for all $w \in F(X)$ with $\pi(w) \in B$, $\Psi(\iota(w)) = \Psi(t^{-1}) \circ \Psi(\iota(v)) \circ \Psi(t)$, where v is any element in the preimage of $\varphi^{-1}(\pi(w))$ under π. Thus indeed $\ker(\pi') \subseteq \ker(\Psi)$, and we obtain $\psi : G^{*(\varphi)} \to \mathcal{S}_{\mathcal{N}}$ by factoring Ψ through $G^{*(\varphi)}$.
What is ψ good for, considering our original intent to show uniqueness of normal form representations? Fix any normal form $(g_0, t^{\epsilon_1}, \ldots, t^{\epsilon_n}, g_n)$. Now it is not difficult to see that the permutation $\psi(\beta(g_0)\pi'(t)^{\epsilon_1} \cdots \pi'(t)^{\epsilon_n}\beta(g_n))$ maps the normal form (1_G) to $(g_0, t^{\epsilon_1}, \ldots, t^{\epsilon_n}, g_n)$, showing that the canonical assignment of elements of $G^{*(\varphi)}$ to normal forms must be injective, which is just what we wanted. □

In the existence proof of finite group presentations with undecidable word problem, we shall mainly be interested in a certain type of HNN extensions:

Definition 3.3.4. *Let G be a group, $H \leqslant G$. Then $G^{*(\mathrm{id}_H)}$ is called the* ***special HNN extension of G w.r.t. H.***

We will see why these are important for us in an instant. First, we need another definition generalizing the notion of "word problem":

Definition 3.3.5. *Let X be a countable alphabet, $R, S \subseteq (X \cup X^{-1})^*$, and fix a well-order $\lhd$ on X. Then the* ***generalized word problem for the presentation $\langle X \mid R\rangle$ w.r.t. S and $\lhd$,*** *denoted $\mathrm{GWP}_{X,R,S,\lhd}$, is the set of all word Gödelizations under $\lhd$ which represent members of the subgroup generated by the elements represented by the words in S.*

Obviously, $\mathrm{WP}_{X,R,\lhd} = \mathrm{GWP}_{X,R,\varnothing,\lhd}$. Again, one can show that in the case of finitely generated groups, the decidability of the generalized word problem does not depend on the exact choice of presentation (as long as we always use finitely many generators), well-order, and, of course, not on the choice of the "generating subset" S. More precisely: If $\varphi : \langle X \mid R\rangle \to \langle Y \mid S\rangle$ is an isomorphism of groups, $|X \cup Y| < \aleph_0$, $\lhd_X$ a well-order on X, $\lhd_Y$ one on Y, $A \subseteq (X \cup X^{-1})^*$, and $B \subseteq (Y \cup Y^{-1})^*$ such that the elements of the second presentation represented by the words in B generate the subgroup which is the image of the subgroup "generated by A" under φ, then $\mathrm{GWP}_{X,R,A,\lhd_X}$ is decidable if and only if $\mathrm{GWP}_{Y,S,B,\lhd_Y}$ is decidable.
Now special HNN extensions allow us to reduce generalized word problems to word problems:

Lemma 3.3.6. *(Our raison d'être for special HNN extensions)*
Let G be a group, $H \leqslant G$. Then for all $g \in G \leqslant G^{*(\mathrm{id}_H)}$, denoting the stable letter of the HNN extension by t: $t^{-1}gt = g \Leftrightarrow g \in H$. In particular, if G is finitely generated and the generalized word problem w.r.t. H in G is undecidable, then the word problem in the finitely generated group $G^{*(\mathrm{id}_H)}$ is undecidable.

Proof. The "⇐" direction of the equivalence is clear. Suppose conversely that $g \in G \backslash H$. Then (g^{-1}, t^{-1}, g, t) is a reduced sequence and thus by Theorem 3.3.3(1), we have $g^{-1}t^{-1}gt \neq 1$ in $G^{*(\mathrm{id}_H)}$.
For the "in particular", note first that clearly, if G is finitely generated, so is $G^{*(\mathrm{id}_H)}$. Now by the above equivalence, fixing a presentation $G = \langle X \mid R\rangle$ with $|X| < \aleph_0$, the map

$(X \cup X^{-1})^* \to (X \cup X^{-1} \cup \{t, t^{-1}\})^*$ given by $w \mapsto w^{-1}t^{-1}wt$ (or actually, more precisely the corresponding map of word Gödelizations) is a reduction of the generalized word problem w.r.t. H in G to the word problem in $G^{*(\mathrm{id}_H)}$. □

3.4 Free products with and without amalgamation

We begin with the following lemma:

Lemma 3.4.1. *Let $(G_i)_{i\in I}$ be a family of groups. Fix two families of presentations $(\langle X_i \mid R_i\rangle)_{i\in I}, (\langle Y_i \mid S_i\rangle)_{i\in I}$, w.l.o.g. $X_i \cap X_j = \varnothing = Y_i \cap Y_j$ for $i \neq j$, such that for all $i \in I$: $\langle X_i \mid R_i\rangle \cong G_i \cong \langle Y_i \mid S_i\rangle$. Then $\langle \bigcup_{i\in I} X_i \mid \bigcup_{i\in I} R_i\rangle \cong \langle \bigcup_{i\in I} Y_i \mid \bigcup_{i\in I} S_i\rangle$.*

Proof. Fix isomorphisms $\psi_i : \langle X_i \mid R_i\rangle \to \langle Y_i \mid S_i\rangle$, $i \in I$. Define a homomorphism $\varphi : F(\bigcup_{i\in I} X_i) \to \langle \bigcup_{i\in I} Y_i \mid \bigcup_{i\in I} S_i\rangle$ on generators by sending $x \in X_i$ to the element represented by some word which represents $\psi_i(x)$ in $\langle Y_i \mid S_i\rangle$. It is easy to check that $\langle\langle \bigcup_{i\in I} R_i\rangle\rangle \subseteq \ker(\varphi)$ so that φ factors to a homomorphism $\psi : \langle \bigcup_{i\in I} X_i \mid \bigcup_{i\in I} R_i\rangle \to \langle \bigcup_{i\in I} Y_i \mid \bigcup_{i\in I} S_i\rangle$. Using the inverses ψ_i^{-1} of the given isomorphisms ψ_i, one can analogously construct a homomorphism ψ^{-1} in the other direction, and it is easy to check that they are inverse; e.g., for $x \in X_i$: $\psi^{-1}(\psi(x)) = \psi^{-1}(w(y_1^{(i)}, \dots, y_m^{(i)})) = w(v_1^{(i)}, \dots, v_m^{(i)})$, where w is a word over Y_i representing $\psi_i(x)$, and $v_j^{(i)}$ is a word over X_i representing $\psi_i^{-1}(y_j^{(i)})$. Thus, in $\langle X_i \mid R_i\rangle$, this word represents $w(\psi_i^{-1}(y_1^{(i)}), \dots, \psi_i^{-1}(y_m^{(i)})) = \psi_i^{-1}(w(y_1^{(i)}, \dots, y_m^{(i)})) = \psi_i^{-1}(\psi_i(x)) = x$. But since all relators from R_i are also among the relators of $\langle \bigcup_{i\in I} X_i \mid \bigcup_{i\in I} R_i\rangle$, the relation $w(v_1^{(i)}, \dots, v_m^{(i)}) = x$ also holds there, so $\psi^{-1} \circ \psi$ is identical on generators and thus identical as a whole. Analogously, one shows $\psi \circ \psi^{-1} = \mathrm{id}$. □

This enables us to make the following definition:

Definition 3.4.2. *Let $(G_i)_{i\in I}$ be a family of groups, with presentations $G_i = \langle X_i \mid R_i\rangle$, w.l.o.g. X_i pairwise disjoint. Then the group $*_{i\in I} G_i := \langle \bigcup_{i\in I} X_i \mid \bigcup_{i\in I} R_i\rangle$ is called the **free product of the** G_i, $i \in I$.*

In the proof of Lemma 3.4.1, we combined the different isomorphisms ψ_i starting on the single G_i to one big isomorphism ψ on the free product. However, the isomorphism property was only used to construct the big inverse isomorphism from the small inverse isomorphisms; the argument of combining works even when the ψ_i are just homomorphisms, giving us the so-called universal property of free products:

Proposition 3.4.3. *Let $(G_i)_{i\in I}$ be a family of groups, G a group, and for $i \in I$, let $\varphi_i : G_i \to G$ be a homomorphism. Also, consider the canonical homomorphisms $\iota_i : G_i \to *_{i\in I} G_i$, $i \in I$. Then there exists exactly one homomorphism $\varphi : *_{i\in I} G_i \to G$ such that for all $i \in I$: $\varphi \circ \iota_i = \varphi_i$.* □

Letting $G := \prod_{i\in I} G_i$ with the φ_i the canonical embeddings, we get that all ι_i must be injective, so all G_i embed canonically into $*_{i\in I} G_i$. Intuitively, the free product of the G_i is the most general group generated by the G_i as subgroups. This should already give us an idea how normal forms for elements of free products look like:

Definition 3.4.4. *Let $(G_i)_{i\in I}$ be a family of groups, w.l.o.g. all G_i disjoint as sets. A finite sequence $(g_1,\dots,g_n)$, $n \geqslant 0$, such that for all $k = 1,\dots,n$ there exists an $i_k \in I$ such that $g_k \in G_{i_k}\backslash\{1\}$ and neighboring elements are not from the same G_i is called* ***free product reduced*** *or also a* ***free product normal form.***

We can now formulate and prove the normal form theorem for free products:

Theorem 3.4.5. *Let $(G_i)_{i\in I}$ be a family of groups. Then:*
(1) If $(g_1,\dots,g_n)$, $n > 0$, is a reduced sequence, then $g_1 \cdots g_n \neq 1$ in $*_{i\in I}G_i$.
(2) Every element of the free product $*_{i\in I}G_i$ has a unique normal form representation.

Proof. The proof is very similar in spirit to the one of Theorem 3.3.3, but will be much shorter because we leave most of it as an exercise to readers who now want to try proving a normal form theorem themselves and because for ease of notation we will not spell out all identifications that we make explicitly any more.
Again, we shall use the implication "(2)⇒(1)", which is immediate considering the convention that the empty normal form represents the identity element.
For (2), first observe that existence of normal form representations is clear, considering the fact that the canonical images of the G_i by definition generate the free product. Now define the action of $*_{i\in I}G_i$ on the set $\mathcal{N}$ by combining actions of the single G_i on $\mathcal{N}$ via the universal property. The action of G_i on $\mathcal{N}$ is defined as follows as a homomorphism $\varphi_i : G_i \to \mathcal{S}_{\mathcal{N}}$: Fix $g \in G_i$. If $g = 1$, then of course let $\varphi_i(g) := id_{\mathcal{N}}$. Else, let $(g_1,\dots,g_n)$ be a normal form, and define $\varphi_i(g)((g_1,\dots,g_n)) := \begin{cases} (g,g_1,\dots,g_n) & \text{if } g_1 \notin G_i \\ (gg_1,g_2,\dots,g_n) & \text{if } g_1 \in G_i\backslash\{g^{-1}\} \\ (g_2,\dots,g_n) & \text{if } g_1 = g^{-1} \end{cases}$.
Now first check that this really defines a permutation of $\mathcal{N}$ (what will be the inverse permutation?) and that this assignment is a homomorphism. Now combine the φ_i, $i \in I$ to a single action φ of the free product on $\mathcal{N}$. Conclude as in the proof of Theorem 3.3.3. □

Finally, we turn to free products with amalgamation, a construction generalizing free products. First, we define:

Definition 3.4.6. *Let $(G_i)_{i\in I}$ be a family of groups, A a group. A* ***group amalgam of the family*** *$(G_i)_{i\in I}$* ***over*** *A is a family $(\alpha_i)_{i\in I}$ of monomorphisms $\alpha_i : A \to G_i$.*

So in a group amalgam, A can be considered a subgroup of all the G_i, $i \in I$, but it is also important which subgroup of G_i is viewed as A, since we now want to "glue" the G_i along the different copies of A:

Definition 3.4.7. *Let $(G_i)_{i\in I}$ be a family of groups, $(\alpha_i)_{i\in I}$ an amalgam of the G_i over A. Then the* ***free product of the*** *G_i* ***amalgamated over*** *A* ***via the*** *α_i is defined to be the quotient $*^{(\alpha_i)}_{i\in I}G_i := *_{i\in I}G_i/\langle\langle \alpha_i(a)^{-1}\alpha_j(a) \mid a \in A, i,j \in I\rangle\rangle$.*

Obviously, for $A = \{1\}$, $*^{(\alpha_i)}_{i\in I}G_i = *_{i\in I}G_i$. We want to understand free products with amalgamation better by proving a normal form theorem for them as well. First, the definition:

Definition 3.4.8. *Let $(G_i)_{i\in I}$ be a family of groups, w.l.o.g. all G_i pairwise disjoint, $(\alpha_i)_{i\in I}$ an amalgam of the G_i over A.*
(1) A finite sequence $(c_1,\dots,c_n)$, $n \geqslant 0$, is called ***amalgam reduced*** *if and only if each c_k is in one of the factors G_i, consecutive c_k are from different factors, and if $n > 1$, none of the c_k is in one of the images of A under the homomorphism α_i, whereas if $n = 1$, $c_1 \neq 1$.*

(2) For each $i \in I$, choose a set $\mathcal{A}_i$ of right coset representatives of $\alpha_i(A)$ in G_i such that $1_{G_i} \in \mathcal{A}_i$. Then a finite sequence $(a, g_1, \ldots, g_n)$, $n \geqslant 0$, is called an ***amalgam normal form w.r.t. the*** *$\mathcal{A}_i$ if and only if $a \in A$, each g_k is in one of the $\mathcal{A}_i \backslash \{1_{G_i}\}$, and neighboring g_k come from different factors.*

Now we can prove:

Theorem 3.4.9. *(Normal Form Theorem for Free Products with Amalgamation)*
Let $(G_i)_{i\in I}$ be a family of groups, w.l.o.g. all G_i pairwise disjoint, $(\alpha_i)_{i\in I}$ an amalgam of the G_i over A, $\mathcal{A}_i$ a set of right coset representatives of $\alpha_i(A)$ in G_i such that $1_{G_i} \in \mathcal{A}_i$, $i \in I$.
(1) If $(c_1, \ldots, c_n)$, $n \geqslant 1$, is a reduced sequence, then in $\ast_{i\in I}^{(\alpha_i)} G_i$: $c_1 \cdots c_n \neq 1$. In particular, the G_i embed canonically into $\ast_{i\in I}^{(\alpha_i)} G_i$.
(2) Each element of $\ast_{i\in I}^{(\alpha_i)} G_i$ has a unique normal form representation.

Proof. The proof is also very similar to the one of Theorem 3.3.3, so we only present the rough ideas and leave the rest as an exercise.
First observe that "(2)⇒(1)" by transforming each reduced form $(c_1, \ldots, c_n)$ into a normal form $(a, g_1, \ldots, g_n)$ representing the same element by "working from the right to the left", using the fact that elements in the different copies of A can be "shifted" from one factor to any other by the relations added in Definition 3.4.7.
So it remains to show (2). First note that existence of normal form representations is again easy, using the existence of reduced form representations (which is clear) and the method to transform reduced forms into normal forms hinted at in the last paragraph. To show uniqueness, define an action of the amalgamated product on the set $\mathcal{N}$ of normal forms; the action will be obtained by factoring an action of the free product of the G_i on $\mathcal{N}$, so just define the actions of the single G_i on $\mathcal{N}$ by "simulation of left multiplication"; more precisely: For $g \in G_i$ and a normal form $(a, g_1, \ldots, g_n)$, first write, in G_i, $g\alpha_i(a) = \alpha_i(a')g'$ with $a' \in A$ and $g' \in \mathcal{A}_i$. In case $g_1 \in G_i$, also write $g'g_1 = \alpha_i(a'')g_1'$ with $a'' \in A$ and $g_1' \in \mathcal{A}_i$. Now define $g \cdot (a, g_1, \ldots, g_n) := \begin{cases} (a', g', g_1, \ldots, g_n), & \text{if } g_1 \notin G_i \text{ and } g' \neq 1_{G_i}, \\ (a', g_1, \ldots, g_n) & \text{if } g_1 \notin G_i \text{ and } g' = 1_{G_i}, \\ (a'a'', g_1', g_2, \ldots, g_n), & \text{if } g_1 \in G_i \text{ and } g_1' \neq 1_{G_i}, \\ (a'a'', g_2, \ldots, g_n) & \text{else} \end{cases}$, and check that this definition "works" (i.e., ascertain that the rest of the proof can be carried out in analogy to the one of Theorem 3.3.3). □

We note that the special case of amalgamated free products with two factors in Theorem 3.4.9 can be reduced to the normal form theorem for HNN extensions (see the proof in (10) on p.187), but in order to apply this idea in the general case, more general types of HNN extensions would be necessary: Let G, a subgroup $H \leqslant G$ and a family $(H_i)_{i\in I}$ of subgroups of G all isomorphic to H be given; further assume that explicit isomorphisms $\alpha_i : H \to H_i$ are given. Then it is possible to construct an extension of G in which for each $i \in I$, there is an element t_i such that α_i is the restriction of conjugation by t_i to H. We will not need this generalization in this thesis, though.

3.5 Nielsen-reducedness

This concept, which falls into the big area of cancellation theory, is a sufficient property for a set of elements of some free group $F(X)$ to be a free basis for the subgroup they generate.

We call $U \subseteq F(X)$ **inverseless** if and only if for each $u \in U$, the formal inverse $u^{-1} \notin U$. In particular, no inverseless U can contain the empty word 1.

Definition 3.5.1. *Let X be an alphabet, $U \subseteq F(X)$ inverseless such that for all $v_1, v_2, v_3 \in U^{\pm 1}$, represented as reduced words over X, the following hold (where, for an element $v \in F(X)$, $|v|$ denotes the length of the reduced form representation of v as a word over $X \cup X^{-1}$):*
(1) $v_2 \neq v_1^{-1} \Rightarrow |v_1 v_2| \geqslant \max\{|v_1|, |v_2|\}$,
(2) $v_2 \neq v_1^{-1} \wedge v_3 \neq v_2^{-1} \Rightarrow |v_1 v_2 v_3| > |v_1| - |v_2| + |v_3|$.
*Then U is called **Nielsen-reduced**.*

Some remarks about this definition: Condition (1) says that in any product of two elements of $U^{\pm 1}$ that are not inverse to each other, at most half as many pairs xx^{-1} or $x^{-1}x$ as there are symbols in the shorter one of the two words must be cancelled to obtain the reduced form. Also, according to (2), whenever three elements $v_1, v_2, v_3 \in U^{\pm 1}$ such that v_2 is not inverse to v_1 and v_3 not to v_2 are concatenated, when passing to the reduced form the middle word v_2 will never be fully cancelled.

Proposition 3.5.2. *Let X be an alphabet, $U \subseteq F(X)$ Nielsen-reduced. Then for each $u \in U^{\pm 1}$, represented as a reduced word over $X \cup X^{-1}$, there are words $a(u), m(u) \in (X \cup X^{-1})^*$ with $m(u) \neq 1$ such that $u = a(u)m(u)a(u^{-1})^{-1}$ in $(X \cup X^{-1})^*$ and such that in each product $w = u_1 \cdots u_t$ with $t \geqslant 0$, $u_i \in U^{\pm 1}$ and $u_{i+1} \neq u_i^{-1}$, no symbols in the middle parts $m(u_1), \ldots, m(u_t)$ are cancelled when passing to the reduced form of w. In particular, $|w| \geqslant t$, and hence U is a free basis for $\langle U \rangle_{F(X)}$.*

Proof. For $u \in U^{\pm 1}$, $a(u)$ is defined as the longest initial segment of u such that there exists $v \in U^{\pm 1}$, $v \neq u^{-1}$, such that in passing to the reduced form of vu, $a(u)$ is cancelled. Note that $a(u^{-1})^{-1}$ is a terminal segment of u, and by (2), $|a(u)| + |a(u^{-1})^{-1}| < |u|$, so that the middle segment $m(u)$ with $u = a(u)m(u)a(u^{-1})^{-1}$ is well-defined and non-empty.
Now consider w as above, and let w' be the reduced form of w. Then by definition, in passing from w to w', the symbols from the $m(u_i)$, $i = 1, \ldots, t$, are never cancelled, q.e.d. The "In particular" clearly follows from the first statement. □

3.6 An embedding theorem

Combinatorial group theory has provided a number of interesting embedding theorems. We will need the following, proved by Higman, Neumann and Neumann in 1949:

Theorem 3.6.1. *Any countable group C can be embedded into a two-generator group G such that if C is **recursively presented** (i.e., C has a presentation of the form $\langle X \mid R \rangle$ where X is finite and $R \subseteq (X \cup X^{-1})^*$ is recursively enumerable), then so is G (with respect to the two generators).*

Proof. Fix a presentation $C = \langle c_1, c_2, \ldots \mid s_1, \ldots \rangle$ of C with countably many generators. Set $F := C * F(a, b)$. Then it is not difficult to see that the set $U := \{b^{-n} a b^n \mid n \in \mathbb{N}\} \subseteq F(a, b)$ is Nielsen-reduced and hence a free basis for the subgroup of $F(a, b)$ that it generates. Similarly, the set $\{b\} \cup \{c_n a^{-n} b a^n \mid n \in \mathbb{N}, n \geqslant 1\}$ is a free basis for the subgroup of F that it generates, because the image of this set under the projection $\pi : F \to F(a, b)$ sending $a \mapsto a, b \mapsto b, c_i \mapsto 1$ for all i is Nielsen-reduced.
It follows that the group $G := \langle F, t \mid t^{-1} a t = b, t^{-1} b^{-i} a b^i t = c_i a^{-i} b a^i, i \geqslant 1 \rangle$ is an HNN extension of F, whence C embeds into G. Also, G is clearly generated by a and t. Note

that the above definition of G actually is an abbreviation for a presentation whose set of generators is $\{c_n \mid n \in \mathbb{N}, n \geqslant 1\} \cup \{t\}$. From this, it is clear that if the presentation of C which we fixed has a finite number of generators and a recursively enumerable set of relators, then so does the presentation of G by which we defined it, and also the one where we remove the superfluous generators b and $c_1, \ldots, c_n$, replacing each occurrence of one of them in the relations by a fixed corresponding word over a, t, a^{-1}, t^{-1}. □

4 Undecidability of the Word Problem

4.1 The Higman Embedding Theorem

This chapter is based on (10); as announced earlier, we will not only establish the existence of finitely presented groups with undecidable word problem, but also prove a powerful result which has many interesting consequences:

Theorem 4.1.1. *(Higman Embedding Theorem)*
Let G be a finitely generated group. The following are equivalent:
(1) G can be embedded into some finitely presented group.
(2) G is recursively presented.

Just as in the case of Matiyasevich's Theorem, one direction of this equivalence, although maybe not completely obvious, is not difficult to see: If a finitely generated group G embeds into a finitely presented group H, say with generators $x_1, \ldots, x_n$ and relators $r_1, \ldots, r_m$, then one can write the finitely many generators of G as words over $x_1, \ldots, x_n, x_1^{-1}, \ldots, x_n^{-1}$. Then one can systematically consider all the elements of the normal subgroup of $F(x_1, \ldots, x_n)$ generated by $r_1, \ldots, r_m$ as well as the elements of $F(x_1, \ldots, x_n)$ obtained by all possible concatenations of the words that represent the generators of G and the formal inverses of these words. Whenever two of these coincide (as elements of $F(x_1, \ldots, x_n)$), one has found a relator for the generators of G, and it is clear that all relators for the generators of G can be obtained this way. However, by the way we described this "algorithm" to find the relators, it is clear that the set they form is recursively enumerable (invoking the Church-Turing-Thesis). This shows "(1)⇒(2)" in Theorem 4.1.1.
Thus, again, we are left with only one "interesting" direction whose proof requires more work. Luckily, we already have Matiyasevich's Theorem at our disposal, which will help us a lot here.

4.2 Benignity and the Higman Rope Trick

The following concept was introduced by Higman to reduce the proof of his embedding theorem to showing a certain statement about normal subgroups of free groups:

Definition 4.2.1. *Let G be a finitely generated group, $H \leqslant G$. H is called* ***benign in*** *G if and only if the special HNN extension G_H embeds into a finitely presented group.*

The next lemma, which can be considered as the raison d'être for benignity, actually only makes use of a consequence of benignity, namely: If H is benign in G, then the free product with amalgamation $G *_H G$ embeds into a finitely presented group. This is true because G_H contains a copy of $G *_H G$, namely the subgroup L generated by G and its conjugate $t^{-1}Gt$; to see this, first observe that clearly, by the HNN relations of G_H, the elements of the form h and $t^{-1}ht$ for $h \in H$ in the two copies are identified so that $\langle G, t^{-1}Gt\rangle_{G_H}$ certainly is a quotient of $G *_H G$, i.e., we have a canonical epimorphism $\varphi : G *_H G \to \langle G, t^{-1}Gt\rangle_{G_H}$. But this epimorphism also has trivial kernel; for this, fix sets $\mathcal{A}_1, \mathcal{A}_2$ of right coset representatives of H in G (viewed each as being a subset of one of the two factors of the free product with amalgamation) that each contain 1_H. Then any nontrivial element $G *_H G$ can be represented by a nonempty amalgam normal form $(h, g_1, \ldots, g_n)$, where $h \in H$ and the g_i are nontrivial representatives alternatingly from $\mathcal{A}_1$ and $\mathcal{A}_2$ (where g_1 may be in either of them). But now we

see that the image of this element under φ has a nontrivial HNN normal form representation, namely either $(hg_1, t^{-1}, g_2, t, g_3, \ldots)$ (in case $g_1 \in \mathcal{A}_1$) or $(h, t^{-1}, g_1, t, g_2, t^{-1}, g_3, t, \ldots)$ (in case $g_1 \in \mathcal{A}_2$), so it is also nontrivial. Now Lemma 4.2.2 shows that in order to prove the Higman Embedding Theorem, it suffices to show that recursively enumerable (normal) subgroups of a free group are benign in that free group:

Lemma 4.2.2. *(The Higman Rope Trick)*
Let F be a finitely presented group, $R \trianglelefteq F$ benign in F. Then F/R embeds into a finitely presented group.

Proof. Fix a finite presentation $\langle X \mid S \rangle$ for F, set $L := F *_R \overline{F}$, where $\overline{F}$ is an isomorphic copy of F with all letters "barred" and let H be a finitely presented group into which L embeds. Fix a third isomorphic copy $\tilde{F}$ of F, where all letters are "tilded". For the "rope trick", we consider the homomorphism $\varphi : L \to \tilde{F}/\tilde{R}$ defined on the first factor F by $w \mapsto \tilde{w}$ and on the second factor $\overline{F}$ by $\overline{w} \mapsto 1$. Note that φ is well-defined because the two definitions agree on the copies of R. Now form the group $H \times \tilde{F}/\tilde{R}$ and observe that L embeds in two ways into this group, namely by the map $l \mapsto (l, 1)$ and by $l \mapsto (l, \varphi(l))$. Hence one can form the HNN extension $K := \langle H \times \tilde{F}/\tilde{R}, s \mid s^{-1}(l, 1)s = (l, \varphi(l)), l \in L \rangle$. Clearly, $\tilde{F}/\tilde{R}$ embeds into K, so we are done once we have shown that K is finitely presented.
Now, a priori, to present K, apart from the stable letter s, we need the generators and relators from a finite presentation of H where w.l.o.g. we can assume that the generators $x, \overline{x}$ $(x \in X)$ for L are included, the "tilded" generators $\tilde{x}$ for $x \in X$, with the finitely many relators for $\tilde{F}$, the (possibly infinitely many) relators from $\tilde{R}$, the finitely many commutation relators between the generators of H and $\tilde{F}$ and finally the HNN relators of the form $s^{-1}ls = l \cdot \Phi(l)$, where l runs through the finite set $X \cup \overline{X}$ of generators of the subgroup $L \leqslant H$ and $\Phi(l)$ is the "canonical" representation of $\varphi(l)$ as a word over the generators, i.e., $\Phi(l) = \tilde{l}$ if $l \in X$ and $\Phi(l) = 1$ if $l \in \overline{X}$. Hence it suffices to show that the relators from $\tilde{R}$ already follow from the other relators specified.
So let $\tilde{w} \in \tilde{R}$. We need to derive the triviality of $\tilde{w}$. On the one hand, $sws^{-1} = w\tilde{w}$ from the first type of HNN relators added. On the other hand, since all relations that hold in $F *_R \overline{F}$ must also follow from the relations between the generators of H, we have $w = \overline{w}$, but also $s \cdot \overline{w} \cdot s^{-1} = \overline{w}$ from the second type of HNN relators. Comparing the two results, $\tilde{w} = 1$ follows. □

So in order to prove the Higman Embedding Theorem, we should come to better understand the concept of benignity; the following lemma summarizes some key facts:

Lemma 4.2.3. *Let G be a finitely generated group embedding into some finitely presented group. Then the following hold:*
(1) Every finitely generated subgroup of G is benign in G.
(2) If $H, K \leqslant G$ are benign in G, then so are $H \cap K$ and $\langle H \cup K \rangle_G$.

Proof. For (1): Let M be a finitely presented group into which G embeds. Then by the normal form theorem for HNN extensions, G_H embeds into M_H, which is also a finitely presented group.
For (2): Suppose G_H embeds into M and G_K into N, where M and N are finitely presented groups; form the free product with amalgamation $P := M *_G N$ and note that P is still finitely presented. By an application of the normal form theorem for free products with amalgamation, we also get that P contains a copy of $G_H *_G G_K$, and this latter group has a

presentation of the form $\langle G, \overline{G}, t, s \mid t^{-1}ht = h\ (h \in H)\ , s^{-1}\overline{k}s = \overline{k}\ (\overline{k} \in \overline{K})\ , g = \overline{g}\ (g \in G)\rangle = \langle G, t, s \mid t^{-1}ht = h\ (h \in H)\ , s^{-1}ks = k\ (k \in K)\rangle$. Now either by the more general theory of HNN extensions with several stable letters hinted at before or simply by viewing this group as obtained from G via two successive special HNN extensions, we get a natural notion of "stable letter reduction" and "reduced form" for this presentation, and an element will be in G if and only if in any sequence representing it we can eventually delete all occurrences of the stable letters s, t and their inverses by single reduction steps.
We can now show that P (or even the subgroup of P generated by G, t and s which we just discussed) contains a copy of $G_{H\cap K} = \langle G, u \mid u^{-1}gu = g\ (g \in G)\rangle$ - namely the subgroup generated by G and the element $(ts)^2$. For this, first observe that conjugation by u fixes all elements from $H \cap K$ so that this subgroup is a quotient of $G_{H\cap K}$. But the corresponding canonical epimorphism also has trivial kernel; just take a nontrivial element from $G_{H\cap K}$, represented by a nontrivial HNN reduced form, say $(g_0, u^{\epsilon_1}, g_1, \ldots, u^{\epsilon_n}, g_n)$. Then under the canonical epimorphism, this element will be mapped to an element of $\langle G, t, s\rangle_P$ represented by the (not necessarily reduced) sequence $(g_0, x_{\epsilon_1}^{\epsilon_1}, y_{\epsilon_1}^{\epsilon_1}, x_{\epsilon_1}^{\epsilon_1}, y_{\epsilon_1}^{\epsilon_1}, g_1, \ldots, x_{\epsilon_n}^{\epsilon_n}, y_{\epsilon_n}^{\epsilon_n}, x_{\epsilon_n}^{\epsilon_n}, y_{\epsilon_n}^{\epsilon_n}, g_n)$, where $x_1 := y_{-1} := t$ and $x_{-1} := y_1 := s$. Now since we started with a reduced sequence in the sense of the special HNN extension $G_{H\cap K}$, it is not difficult to see that when performing stable letter reductions (with respect to s and t, of course) on this sequence to obtain a reduced representation of the image, in each of the four-letter blocks obtained from the occurrences of u and u^{-1}, the two central symbols will never be cancelled. But then by Britton's Lemma, this image is also nontrivial, q.e.d.
To show benignity of $\langle H \cup K\rangle_G$ in G, we claim that $\langle H \cup K\rangle_G = \langle s^{-1}Gs, t^{-1}Gt\rangle_P \cap G$ so that by what we already know, $\langle H \cup K\rangle_G$ is benign in P and hence in G.
The inclusion "$\subseteq$" for the equality which we want to show is clear; for the other, assume we are given an element of $\langle s^{-1}Gs, t^{-1}Gt\rangle_P \cap G$. Then this element is a product of elements of the form $t^{-1}gt$ and $s^{-1}g's$, $g, g' \in G$, and because it also in G, there is a sequence of stable letter reduction steps that eventually removes all stable letter occurrences (without changing the other entries of the sequence). But by an easy induction on the number of factors of such a product, we see that the product of the elements of G appearing between the stable letters, which coincides with the element itself, is in $\langle H \cup K\rangle_G$, q.e.d. □

4.3 Valiev's principal lemma and finitely presented groups with undecidable word problem

Recall that our final goal is to show that recursively enumerable subgroups of finitely generated free groups are benign. In this section, we follow a proof by Valiev of a special case of this statement, from which with some extra work (see the next section) one can derive the general case:

Lemma 4.3.1. *(Principal Lemma, Valiev 1968)*
Let $S \subseteq \mathbb{Z}$ be recursively enumerable. Then the subgroup of $F(a_0, b_0, c_0)$ generated by elements of the form $a_0^n b_0 c_0^n$ with $n \in S$ is benign in $F(a_0, b_0, c_0)$.

Proof. By Matiyasevich's Theorem, fix a polynomial $P(X_0, \ldots, X_t)$ such that for all $z_0 \in \mathbb{Z}$: $z_0 \in S \Leftrightarrow \exists z_1, \ldots, z_t \in \mathbb{Z}[P(z_0, \ldots, z_t) = 0]$. Note that by proceeding recursively over the construction of $P(X_0, \ldots, X_t)$ as a term in the signature $\mathbb{Z} \cup \{+, \cdot\}$ and introducing new variables, we can transform the above equivalence to one of the form $z_0 \in S \Leftrightarrow \exists z_1, \ldots, z_m \in$

$\mathbb{Z} : \mathcal{M}(z_0, \ldots, z_m)$, where $\mathcal{M}(z_0, \ldots, z_m)$ is a conjunction of atomic formulas over this signature, which are each of one of the forms $X_i = c$ (with $c \in \mathbb{Z}$ an integer constant), $X_i = X_j$, $X_i + X_j = X_k$ (with i, j, k distinct) or $X_l = X_i \cdot X_j$ (with $0 < l < i < j \leqslant m$; this restriction, which can be achieved by keeping the indices of the variables of P and successively increasing indices in the variables newly introduced when "working through" the monomials one after another, is for technical reasons that will reveal themselves later).

Having split up the polynomial characterization for z_0 to be in S into many "small parts", we now argue how this reduces the statement to the assertion that certain special subgroups of the free group $F := F(a_0, b_0, c_0, \ldots, a_m, b_m, c_m)$ are benign. For this, associate to each tuple $(z_0, \ldots, z_m) \in \mathbb{Z}^{m+1}$ a "code word" $w_{(z_0,\ldots,z_m)} \in F$ via

$$w_{(z_0,\ldots,z_m)} := c_m^{-z_m} b_m^{-1} a_m^{-z_m} \cdots c_1^{-z_1} b_1^{-1} a_1^{-z_1} a_0^{z_0} b_0 c_0^{z_0} a_1^{z_1} b_1 c_1^{z_1} \cdots a_m^{z_m} b_m c_m^{z_m}.$$

The "right way" to look at this definition is the following: The code words $w_{(z_0,\ldots,z_m)}$ consist of a "sensible" middle part $a_0^{z_0} b_0 c_0^{z_0}$ that is "protected" by the rest of the word, split up into an initial segment and a terminal segment that coat it. Guided by this intuition, it is not difficult to see that the set $U := \{w_{(z_0,\ldots,z_m)} \mid (z_0, \ldots, z_m) \in \mathbb{Z}^{m+1}\}$ is Nielsen-reduced. Also, setting $A_{\mathcal{M}} := \langle w_{(z_0,\ldots,z_m)} \mid \mathcal{M}(z_0, \ldots, z_m)\rangle_F$, we have $\langle a_0^{z_0} b_0 c_0^{z_0} \mid z_0 \in S\rangle_{F(a_0,b_0,c_0)} = \langle A_{\mathcal{M}}, a_1, b_1, c_1, \ldots, a_m, b_m, c_m\rangle_F \cap \langle a_0, b_0, c_0\rangle_F$; to see "$\supseteq$", just observe that in any concatenation of the specified generators of the RHS and their inverses, the word over the generators a_0, b_0, c_0 and their inverses that we get by ignoring all other generators appearing represents an element of the LHS and that this property is not destroyed by reductions. So by Lemma 4.2.3, we are done once we have shown that $A_{\mathcal{M}}$ is benign in F.

But this task, as announced just before, can be split up into many "small tasks": For each elementary formula Γ mentioned in $\mathcal{M}$, set $A_\Gamma := \langle w_{(z_0,\ldots,z_m)} \mid \Gamma(z_0, \ldots, z_m)\rangle_F$. Then if $\mathcal{M}$ is precisely the conjunction of $\Gamma_1, \ldots, \Gamma_p$, we have $A_{\mathcal{M}} = \bigcap_{i=1}^{p} A_{\Gamma_i}$, where "$\subseteq$" is trivial, and "$\supseteq$" holds because U is a free subset of F. Hence another application of Lemma 4.2.3 shows that we only need to prove benignity in F of the special subgroups A_Γ, where Γ is an elementary formula.

The strategy for showing that all A_Γ are benign is as follows: First, we construct a finitely presented group M that contains F, and then associate to each Γ a finitely generated subgroup $L_\Gamma \leqslant M$ such that $L_\Gamma \cap F = A_\Gamma$. Then we will be done by one last application of Lemma 4.2.3. M will be an HNN extension (with several stable letters) of an HNN extension F^* of F. We make this construction to allow us to manipulate code words with conjugation by the stable letters introduced in the process so that L_Γ, although finitely generated, will contain all code words corresponding to Γ because we are able to get all these infinitely many code words from just one by according manipulations. To be more precise, we want stable letters $t_0, \ldots, t_m$ and $p_{j,l}$ for $0 < l < j \leqslant m$ such that $t_i^{-1} w_{(z_0,\ldots,z_m)} t_i = w_{(z_0,\ldots,z_{i-1},z_i+1,z_{i+1},\ldots,z_m)}$ and $p_{j,l}^{-1} w_{(z_0,\ldots,z_m)} p_{j,l} = w_{(z'_0,\ldots,z'_m)}$ with $z'_l = z_l + z_j$ and $z'_s = z_s$ for $s \neq l$.

We first form F^* by adding new generators $t_0, \ldots, t_m$ and for $i = 0, \ldots, m$ the defining relations $t_i^{-1} b_i t_i = a_i b_i c_i$ and $[t_i, x] = 1$ for all generators x of F distinct from b_i. Since under conjugation by t_i, the canonical free set of generators of F is mapped to another free set of generators (to see this, one can either show combinatorially that whenever $x_1, \ldots, x_n$ is a free basis for the subgroup generated then $x_1 x_2, x_2, \ldots, x_n$ and $x_2 x_1, x_2, \ldots, x_n$ are as well, or that free groups are residually finite, whence finitely generated free groups are Hopfian, from which it follows that any n-element generating subset of a free group over n symbols is a free basis for this group) it is clear that F^* is an HNN extension of F with stable letters $t_0, \ldots, t_m$.

Now for M itself, we add to F^* new generators $p_{j,l}$ for $0 < l < j \leqslant m$ and, for each pair (j,l) that satisfies these inequalities, the defining relations $p_{j,l}^{-1} c_j p_{j,l} = t_l c_j$ as well as $[p_{j,l}, x] = 1$, where x runs through all generators of F distinct from c_j and for $x = t_l$. We now argue that M is an HNN extension of F^* by showing that conjugation with $p_{j,l}$ restricts to an automorphism of the subgroup $\langle F, t_l \rangle_M$, which corresponds to the HNN extension of F using the single stable letter t_l and the HNN relations for t_l included among the HNN relations for F^*.
For this, observe that clearly, the endomorphism $\varphi_{j,l}$ of the free group $E_l := F * F(t_l)$ sending c_j to $t_l c_j$ and all other canonical generators to themselves is an automorphism of E_l. We compose $\varphi_{j,l}$ and $\varphi_{j,l}^{-1}$ with the canonical epimorphism $\pi : E_l \to \langle F, t_l \rangle_M$ and show that the compositions factor through the normal subgroup of E_l generated by the defining relators of $\langle F, t_l \rangle_M$ as an HNN extension of F as just described above; then clearly, the two endomorphisms of $\langle F, t_l \rangle_M$ we obtain correspond to the restriction of the conjugation with $p_{j,l}$ and $p_{j,l}^{-1}$ respectively and are inverse to each other, giving the desired result. Now note that defining relators which do not contain c_j are no problem at all. The only defining relator which does contain c_j is $t_l^{-1} c_j t_l c_j^{-1}$, and this is sent to $t_l^{-1} \cdot t_l c_j \cdot t_l \cdot c_j^{-1} t_l^{-1} = 1$ by $\pi \circ \varphi_{j,l}$ (note that c_j and t_l commute in $\langle F, t_l \rangle_M$) and to $t_l^{-1} \cdot t_l^{-1} c_j \cdot t_l \cdot c_j^{-1} t_l = 1$ by $\pi \circ \varphi_{j,l}^{-1}$.
That the desired relations for manipulating code words by conjugation with stable letters hold follows from easy computations and is left as an exercise (note that you will need the technical restriction from the $\cdot$ case when verifying the second type of manipulation relation).
But now we are almost done; as announced, we associate to each elementary formula Γ a finitely generated subgroup L_Γ of M; more precisely:
(1) We associate to $X_i = c$ the subgroup $L_i^c := \langle w_{(0,\dots,0,c,0,\dots,0)}, t_s, s \neq i \rangle_M$, where in the index of w, c is the i-th entry of the tuple.
(2) We associate to $X_i = X_j$ the subgroup $L_{i,j}^{=} := \langle w_{(0,\dots,0)}, t_s (s \neq i, j), t_i t_j \rangle_M$.
(3) We associate to $X_i + X_j = X_k$ the subgroup $L_{i,j,k}^{+} := \langle w_{(0,\dots,0)}, t_s (s \neq i, j, k), t_i t_k, t_j t_k \rangle_M$.
(4) We associate to $X_i \cdot X_j = X_l$ the subgroup $L_{i,j,l}^{\cdot} := \langle w_{(0,\dots,0)}, t_s (s \neq i, j, l), t_i p_{j,l}, t_j p_{i,l} \rangle_M$.
With these definitions of the different types of L_Γ, it is not difficult to see that $A_\Gamma \subseteq L_\Gamma \cap F$. The converse inclusion follows from an application of Britton's Lemma: Assume that an element $u \in L_\Gamma$ is also contained in F. Then we must be able to transform it into a word $w \in F$ by stable letter reductions. We begin this process with a product of the specified generators in which the index tuple of each code word that appears satisfies Γ. When we make stable letter reductions with respect to a single stable letter added in the definition to the generators of L_Γ (such as t_s for $s \neq i, j$ in case (2)), the corresponding reductions do not destroy this property of index tuples. And when the reduction is with respect to a stable letter that we only added as part of a two-factor product to the generators (such as t_i in $t_i t_j$ in case (2)), then after this reduction, we can immediately also reduce with respect to its "partner", and this "double-reduction" as a whole does not destroy the property. Hence the word $w \in F$ into which we can transform u is a product of generators of A_Γ and hence in A_Γ, q.e.d. □

As a "side-product", we now obtain:

Corollary 4.3.2. *There exists a finitely presented group G with undecidable word problem.*

Proof. Fix a recursively enumerable, but not recursive set S of natural numbers. Such a set exists by Proposition 2.4.5 and Theorem 2.1.4; alternatively, one can invoke Theorem

1.3.12 and use some (in both directions) effectively computable bijection between the set of all strings and the set of string codes of natural numbers.
Set $U := \{a^s bc^s \mid s \in S\} \subseteq F(a,b,c) =: K$. The generalized word problem with respect to $B := \langle U \rangle_K$ in K cannot be decidable, because $s \mapsto a^s bc^s$ is a reduction from S to it, since $\{a^s bc^s \mid s \in \mathbb{N}\} \subseteq K$ is Nielsen-reduced and thus free. Hence by Lemma 3.3.6, the word problem in K_B cannot be decidable. But by Lemma 4.3.1, B is benign in K, i.e., K_B embeds into some finitely presented group $G = \langle X \mid R \rangle$, $|X \cup R| < \aleph_0$, say via ι.
Now fix a presentation $\langle Y \mid R' \rangle$ of K_B with $|Y| < \aleph_0$. For each $y \in Y$, fix a representation $w(y)$ of $\iota(y)$ as a word over the generators from X and their formal inverses. Then the map $(Y \cup Y^{-1})^* \to (X \cup X^{-1})^*, y_1^{\epsilon_1} \cdots y_n^{\epsilon_n} \mapsto w(y_1)^{\epsilon_1} \cdots w(y_n)^{\epsilon_n}$, is a reduction from the word problem in K_B to the word problem in G. Hence the latter cannot be decidable. □

4.4 Proof of the Higman Embedding Theorem

First note that by Theorem 3.6.1, it suffices to show that recursively enumberable subgroups N of $L = F(a,b)$ are benign in L. We assign to each $w \in L$, represented by a reduced word, a code number $\gamma(w) \in \mathbb{N}$, in the following way: By definition, $\gamma(\varnothing) := 0$. For nonempty reduced words w, $\gamma(w)$ is defined to be the number whose 10-ary representation is obtained by replacing in w each symbol a by 1, b by 2, a^{-1} by 3 and b^{-1} by 4. Observe that this assignment $w \mapsto \gamma(w)$ is effectively computable in both directions; that is, there are Turing machines $\mathbb{A}$ and $\mathbb{B}$ such that $\mathbb{A}$ on any input x first decides whether x is the string code of some word $w \in L$ and if so, computes the string code of $\gamma(w)$, whereas $\mathbb{B}$ on x decides whether x is the string code of $\gamma(w)$ for some $w \in L$ and if so, outputs the string code of w.
The code numbers $\gamma(w)$ serve to define for each $w \in L$ a code word $g_w \in F(a,b,c,d,e,h) =: F$ via $g_w := whc^{\gamma(w)}de^{\gamma(w)}$. Note that $\{g_w \mid w \in L\}$ is a Nielsen-reduced subset of F (more specifically, the parts $(hc^{\gamma(w)}d)^\epsilon$ are never touched in the reduction process) and hence freely generates $G := \langle g_w \mid w \in L \rangle_F$.
Now suppose that $N \leqslant L$ is recursively enumerable. In particular, N has a recursively enumerable generating subset X. Now setting $Y := \langle h, a, b, c^i de^i, i \in \gamma(X) \rangle_F$ (where the set of generators specified is Nielsen-reduced, although we will not need this fact) and observing that the reduced form (as an element of $F(a,b,c,d,e,h)$) of any element of Y has the property that any power of c whose exponent is not of the form $\pm\gamma(x)$ for some $x \in X$ appearing in it is surrounded by occurrences of d and d^{-1} (in either order), we obtain that $G \cap Y$ is the subgroup generated by all code words g_x with $x \in X$. Consequently, we get: $N = \langle G \cap Y, h, c, d, e \rangle_F \cap \langle a, b \rangle_F$. By Lemma 4.2.3 and Lemma 4.3.1, Y is benign in F so that if we are able to show that G is benign in F as well, we will be done by Lemma 4.2.3.
To show this, just as in the proof of Lemma 4.3.1, we will form an HNN extension F^* of F where the HNN relations will allow us to manipulate the code words g_w in F^* by appropriate stable letter conjugations. We introduce for each symbol $\lambda \in \{a^{\pm 1}, b^{\pm 1}\}$ a stable letter t_λ, and the defining relations $t_\lambda^{-1} h t_\lambda = \lambda h, [t_\lambda, a] = [t_\lambda, b] = 1, t_\lambda^{-1} c t_\lambda = c^{10}, t_\lambda^{-1} e t_\lambda = e^{10}, t_\lambda^{-1} d t_\lambda = c^{\gamma(\lambda)} d e^{\gamma(\lambda)}$. Note that the image of the free generating subset $\{a,b,c,d,e,h\}$ of F is still free in F, whence F^* really is an HNN extension of F.
From the defining relations for F^*, it is readily checked that the following two statements are valid:
(1) If $w = \lambda_1 \cdots \lambda_n$ is a (reduced) word over $\{a, b, a^{-1}, b^{-1}\}$, then $t_{\lambda_n}^{-1} \cdots t_{\lambda_1}^{-1} h d t_{\lambda_1} \cdots t_{\lambda_n} = g_w$.
(2) If the (reduced) word w over $\{a, b, a^{-1}, b^{-1}\}$ ends in the letter λ, say $w = u\lambda$, then $t_\lambda g_w t_\lambda^{-1} = g_u$.

We claim that $G = \langle hd, t_a, t_{a^{-1}}, t_b, t_{b^{-1}} \rangle_{F^*} \cap F$, from which benignity of G in F^* and thus in F immediately follows. The inclusion "$\subseteq$" is clear by (1).
For the other inclusion, suppose that an element T of $\langle hd, t_a, t_{a^{-1}}, t_b, t_{b^{-1}} \rangle_{F^*}$, written as a freely reduced word over hd and the t_λ, is contained in F. Then there must be a sequence $T = T_0, T_1, \ldots, T_m$ such that for $i = 0, \ldots, m-1$, T_{i+1} is obtained from T_i by a single stable letter reduction in the sense of the HNN extension F^* of F, and T_m does not contain any stable letters. We claim (and are done once we have proved) that for $i = 0, \ldots, m$, the following holds: If z is a subword of T_i appearing between successive stable letter occurrences, then $z \in G$.
We show the claim by induction on i. For $i = 0$, this is clear, since $hd \in G$. For the inductive step, note that by (2), reductions of the form $t_\lambda^{-1} z t_\lambda$ are no problem (we are only manipulating the code words to represent words that have one more letter at the end). However, we do need to justify why reductions of the form $t_\lambda z t_\lambda^{-1}$ do not lead out of G.
So suppose $z \in G$; write $z = g_{w_1}^{\epsilon_1} \cdots g_{w_n}^{\epsilon_n}$ reduced over the g_w. Recall that when we freely reduce this product, the subwords $(hc^{\gamma(w_i)}d)^{\epsilon_i}$ will not be touched by the various cancellation steps, which allows us to still read off the code words (and their exponents) from the original product representation of z from its freely reduced form.
Now if $z \in C_\lambda$, where C_λ is the subgroup of F associated with t_λ, z is a reduced word in the free generators $a, b, c^{10}, c^{\gamma(\lambda)} d e^{\gamma(\lambda)}, e^{10}, \lambda h$ and their formal inverses, which implies that we can write z in reduced form as a word of the form $z = y_1 c^{n_1} y_2 \cdots y_k c^{n_k} y_{k+1}$, where the y_i are reduced words over a, b, d, e, h and their formal inverses, all y_i except the very first and very last are nonempty and each of the n_i is congruent to 0 or $\pm\gamma(\lambda)$ modulo 10; we do so by joining together powers of c and e from neighboring factors and noting that this already gives the normal form of z in the sense of F viewed as a free product of six free groups each over a single generator. From this, it is also clear that none of the y_i contains a successive subword of the form hd (because each occurrence of d in the reduced form is surrounded by nontrivial powers of c and e).
But this shows that none of the w_i from above can be empty or end in a letter different from λ. Hence conjugation by t_λ^{-1} just transforms z into another product of code words, q.e.d. $\square$

5 Decidability of the word problem

We end this thesis with some notable results on classes of (presentations of) groups whose word problem is decidable. In particular, a theorem of Gromov about hyperbolic groups may be seen as an interesting antithesis to what we have dealt with so far.

5.1 One-relator groups

We begin with a classical result due to Magnus on **one-relator groups**, i.e., groups that have a presentation of the form $\langle X \mid r \rangle$, where r is a single relator. The source for this section is (10). The methods introduced by Magnus serve not only to show decidability of the word problem in one-relator groups, but many other important theorems about these groups; we shall illustrate it in detail by means of the following result:

Theorem 5.1.1. *(The Freiheitssatz)*
Let $G = \langle X \mid r \rangle$ be a one-relator group with r cyclically reduced (see the remark right afterward). Then if L is a subset of the set of generators such that at least one of the generators that appear in r is not in L, then L is a free basis for $M := \langle L \rangle_G$.

A remark on terminology: A reduced word $w \in (X \cup X^{-1})^*$ is called **cyclically reduced** if and only if the first and last letter of w are not inverse to each other. Note that each reduced word w can be brought into a cyclically reduced form by appropriate conjugations; this form is in general not unique, but there are only finitely many cyclically reduced forms for each reduced word and they are all of the same length.
Also, for an element $w \in F(X)$ and $x \in X$, we denote by $\sigma_x(w)$ the sum of the exponents of the occurrences of $x^{\pm 1}$ in w (note that this is well-defined in terms of the representatives of the congruence class that w literally is) and call this number the **x-balance of** w.

Proof of Theorem 5.1.1. It suffices to show the statement for L such that $|X \backslash L| = 1$, where the one generator omitted is mentioned in r. The proof is by induction on the length of r. If r is a power of a generator, say $r = x^n$ with $n \in \mathbb{Z} \backslash \{0\}$, then $G \cong (\mathbb{Z}/n\mathbb{Z}) * F(X \backslash \{x\})$, and the statement is clear. Hence we can assume that at least two generators, say t and b, are mentioned in r.
We now make a case distinction that is characteristic for the proofs following Magnus's method. For the first case, let us assume that for some generator appearing in r, say t, we have $\sigma_t(r) = 0$. In this case, we will be able to display G as an HNN extension of another one-relator group whose defining relator s in cyclically reduced form is shorter than r.
W.l.o.g., r starts with $b^{\pm 1}$. The idea is that, in this case and in $F(X)$, r is an element of the normal subgroup generated by the generators different from t. To make this idea formal, for $i \in \mathbb{Z}$, and $x \in X \backslash \{t\}$, set $x_i := t^i x t^{-i} \in F(X)$. We can then, in $F(X)$, write r as a cyclically reduced word s on the various x_i, by replacing each occurrence of some letter $x \in X \backslash \{t\}$ by x_i for i the sum of the exponents of occurrences of t before this occurrence of x (using the fact that $\sigma_t(r) = 0$ so this rewriting will work out even in the end). Note that s is shorter than r because its length coincides with the number of letters different from t and t^{-1} in r.
By assumption, b occurs in r, so we can let μ and m denote the minimum and maximum of all the subscripts of b that are used in the rewriting process; by our "W.l.o.g." from the beginning of this paragraph, $\mu \leqslant 0$ and $m \geqslant 0$.

To show that G is indeed an HNN extension as asserted, we consider the group G^* given by the following presentation:

$$G^* = \langle t, b_\mu, \ldots, b_m, x_i (x \in X \backslash \{t, b\}, i \in \mathbb{Z}) \mid s = 1, t b_j t^{-1} = b_{j+1} (j = \mu, \ldots, m-1),$$

$$t x_i t^{-1} = x_{i+1} (x \in X \backslash \{t, b\}, i \in \mathbb{Z}) \rangle.$$

We claim that G^* is isomorphic to G.
Consider the homomorphism $\varphi : G \to G^*$ defined by $t \mapsto t, x \mapsto x_0 (x \in X \backslash \{t\})$ (which is well-defined because the word r is mapped to the word s) and the homomorphism $\eta : G^* \to G$ defined by $t \mapsto t, x_i \mapsto t^i x t^{-i} (x \in X \backslash \{t\})$ (which is also well-defined; the rewriting process from before shows that the relator s is mapped to 1_G, and preservation of the other relations is trivial). But one easily verifies that φ and η are inverse, proving $G \cong G^*$.
Now set $H := \langle b_\mu, \ldots, b_m, x_i (x \in X \backslash \{t\}, i \in \mathbb{Z}) \mid s = 1 \rangle$, and $A := \langle b_\mu, \ldots, b_{m-1}, x_i (x \in X \backslash \{t, b\}, i \in \mathbb{Z}) \rangle_H$, $B := \langle b_{\mu+1}, \ldots, b_m, x_i (x \in X \backslash \{t, b\}, i \in \mathbb{Z}) \rangle_H$. From the induction hypothesis, it is clear that A and B are freely generated by the indicated generators so that conjugation by t in G^* indeed restricts to an isomorphism $A \to B$, and G is an HNN extension of H.
To end the proof in the first case, distinguish two subcases, depending on whether the generator omitted by L is t or some other generator, w.l.o.g. b.
For the first subcase, note that the elements $x \in L$ under the isomorphism $G \to G^*$ turn into x_0, and that s contains at least one occurrence of x_i for some $x \in L$ and $i \in \mathbb{Z} \backslash \{0\}$ so that by the induction hypothesis, L is free in H and hence in G.
For the second subcase, let w be some nontrivial reduced word over L. It is clear that $w \neq 1$ in G if $\sigma_t(w) \neq 0$. But if $\sigma_t(w) = 0$, the rewriting process that we used in passing from r to s can also be applied to w, writing it as a reduced nontrivial word w^* over $J = \{x_i \mid x \in X \backslash \{t, b\}, i \in \mathbb{Z}\}$. Since J omits all b_i for $i \in \mathbb{Z}$, by the induction hypothesis $w^* \neq 1$ in H and hence in G. This concludes the proof of the theorem in the first case.
In the second case, for all generators x mentioned in r, we have $\sigma_x(r) \neq 0$. t is no longer distinguished from b by a special assumption (we only still assume that both are mentioned in r) so that w.l.o.g., we can assume that $L = X \backslash \{t\}$. We set $\alpha := \sigma_t(r)$ and $\beta := \sigma_b(r)$. We reduce to the first case by considering, for two letters $x, y \notin X$, the group $C := \langle (X \backslash \{t, b\}) \cup \{x, y\} \mid r' \rangle$, where r' is obtained from r by replacing each occurrence of t by $y x^{-\beta}$ and of b by x^α. Then by construction, we can define a homomorphism $\Psi : G \to C$ by $t \mapsto y x^{-\beta}, b \mapsto x^\alpha, c \mapsto c$ $(c \in X \backslash \{t, b\})$.
Now note that $\sigma_x(r') = 0$ and $\sigma_y(r') = \alpha \neq 0$, properties that do not change under reductions. So first cyclically reduce r' such that the result starts with an occurrence of y or y^{-1} and then apply the rewriting process with x in place of t and y in place of b to r' to the cyclically reduced form to obtain a word s over the generators of the form y_i and c_i for $i \in \mathbb{Z}$ and $c \in X \backslash \{t, b\}$ which certainly is shorter than r. Hence we can apply the induction hypothesis to get as before (viewing C as an HNN extension of a one-relator group with relator s) that $(X \backslash \{b, t\}) \cup \{x\}$ is free in C (note that this argumentation even works when x does not occur in the cyclically reduced form of r'). It follows that also $(X \backslash \{b, t\}) \cup \{x^\alpha\}$ is free in C. But Ψ maps L elementwise onto this set so that L must be free in G, q.e.d. □

Now that we have the Freiheitssatz at our disposal, we can "recycle" parts of its proof to turn them into general methods for dealing with one-relator groups:

Lemma 5.1.2. *Let X be an alphabet with $|X| \geqslant 2$, $G = \langle X \mid r \rangle$ a one-relator group with cylically reduced r starting with an occurrence of $b^{\pm 1}$ for some generator b and such that for some generator $t \neq b$, $\sigma_t(r) = 0$. Let s be the result of applying the rewriting procedure described in the proof of Theorem 5.1.1 with stable letter t to r. Let μ and m be the minimum and maximum subscript of b in s; note that $\mu \leqslant 0$ and $m \geqslant 0$. Then G is the HNN extension of the one-relator group $H = \langle b_\mu, \ldots, b_m, x_i \, (x \in X \backslash \{t, b\}, i \in \mathbb{Z}) \mid s = 1 \rangle$ with respect to the isomorphism $A \to B$ of the subgroups $A := \langle b_\mu, \ldots, b_{m-1}, x_i \, (x \in X \backslash \{t, b\}, i \in \mathbb{Z}) \rangle_H$ and $B := \langle b_{\mu+1}, \ldots, b_m, x_i \, (x \in X \backslash \{t, b\}, i \in \mathbb{Z}) \rangle_H$ given by $b_j \mapsto b_{j+1}$ and $x_i \mapsto x_{i+1}$ for $j = \mu, \ldots, m-1$ and $x \in X \backslash \{t, b\}, i \in \mathbb{Z}$. Also, s starts with b_0, and if t or t^{-1} was mentioned in r, then the length of s is shorter than that of r and at least one of the subscripts of symbols occurring in s is different from 0.*

Proof. This is almost literally the same proof as the part of the proof of Theorem 5.1.1 until G was exhibited as an HNN extension, except that when the induction hypothesis is invoked in order to show that this really is an HNN extension, here we invoke Theorem 5.1.1 instead. □

The beginning of the proof for the second case also turns into a lemma, but we can show even a bit more than what we did there:

Lemma 5.1.3. *Let X be an alphabet with $|X| \geqslant 2$, $G = \langle X \mid r \rangle$ a one-relator group with r cyclically reduced such that r contains at least two generators and such that for each generator $c \in X$ that appears in r, we have $\sigma_c(r) \neq 0$. Fix two letters $x, y \notin X$ as well as two generators t, b appearing in r, set $\alpha := \sigma_t(r)$ and $\beta := \sigma_b(r)$ and let r' be the result of replacing in r each occurrence of t by $yx^{-\beta}$ and of b by x^α. Then G embeds into $C := \langle (X \backslash \{t, b\}) \cup \{x, y\} \mid r' \rangle$ via the homomorphism Ψ defined by $t \mapsto yx^{-\beta}, b \mapsto x^\alpha, c \mapsto c \, (c \in X \backslash \{t, b\})$. Furthermore, note that $\sigma_x(r') = 0$, $\sigma_y(r') = \alpha \neq 0$ and let s be the word obtained by cyclically reducing r', cyclically shifting the letters of the cyclically reduced form so that it starts with an occurrence of y or y^{-1} and then applying the rewriting process with stable letter x to r'; observe that s starts with an occurrence of $y_0^{\pm 1}$ and that the length of s is shorter than that of r. Consequently, by an application of Lemma 5.1.2 we can exhibit C as an HNN extension of a one-relator group whose defining relator is shorter than that of G.*

Proof. This is also all clear by the proof of Theorem 5.1.1 except for the claim that Ψ is injective. To see this, observe that by Theorem 5.1.1, b is of infinite order in G, whence the group $C' := \langle X \cup \{x\} \mid r, b = x^\alpha \rangle$ is the free product of G and $F(x)$, amalgamating $b = x^\alpha$. But C' is isomorphic to C via $t \mapsto yx^{-\beta}, b \mapsto x^\alpha, x \mapsto x, c \mapsto c \, (c \in X \backslash \{t, b\})$ because one can transform the two presentations into one another by a finite sequence of so-called Tietze transformations (which do not change the isomorphism type of the group presented) as follows: For passing from C' to C, add a new generator y and the relation $t = yx^{-\beta}$, then delete the generators t and b, replacing their occurrences in r by $yx^{-\beta}$ and x^α respectively. Now the claim follows from the normal form theorem for free products with amalgamation (more precisely, the fact that the canonical maps from the factors into the product are embeddings). □

We can now formulate and prove a theorem which includes decidability of the word problem for one-relator presentations with a countable set of generators as a special case:

Theorem 5.1.4. *Let X be a countable alphabet, $\lhd$ a well-order of X with $\mathrm{type}(\lhd) \leqslant \omega$, $r \in (X \cup X^{-1})^*$ cyclically reduced, and consider the one-relator presentation $\langle X \mid r \rangle$. Let K*

be a subgroup of the group presented, given by a recursive (with respect to the enumeration of X induced by $\lhd$) subset $L \subset X$ as a generating subset for K. Then $\mathrm{GWP}_{X,r,L,\lhd} := \mathrm{GWP}_{X,\{r\},L,\lhd}$ *is decidable.*

Proof. By induction on $|r|$. If r is a power of a generator, say $r = c^n, n \in \mathbb{Z}\backslash\{0\}$, then $G \cong (\mathbb{Z}/n\mathbb{Z}) * F(X\backslash\{c\})$, and we can effectively bring a word $w \in (X \cup X^{-1})^*$ into its normal form with respect to this free product representation; by the normal form theorem for free products, $w \in K$ if and only if each factor from the normal form representation only consists of letters from L, which can be effectively checked using the recursiveness of L.
Hence we can assume that r contains at least two different generators. We make the characteristic case distinction.
First, assume that for some generator occurring in r, say t, $\sigma_t(r) = 0$. As in Lemma 5.1.2, maybe replacing r by an appropriate conjugate (if r starts with $t^{\pm 1}$), we can exhibit G as an HNN extension $G \cong \langle H, t \mid t^{-1}At = B\rangle$ of a one-relator group H with shorter defining relator; note that there are several "natural" choices for a well-order on the generators of this new presentation such that a set of generators from X is recursive if and only if its image among the new generators is recursive and such that the generating sets for A and B are recursive. Hence the induction hypothesis applies to A and B in H; since the isomorphism from G to this HNN extension is also effectively computable, we can algorithmically decide the word problem in G by first computing, from a word $w \in (X \cup X^{-1})^*$, a representation of w in the HNN extension, then effectively performing stable letter reductions using decidability of the generalized word problem with respect to the generating subsets of A and B in H and then checking if the reduced form is empty or not.
There are two subcases to distinguish. First, denote by R the set of all generators from X that occur in r and assume that $R \subseteq L$. Then since G is the free product of $\langle R \mid r\rangle$ with $F(X\backslash R)$ and using the decidability of the word problem in G, we can effectively compute for each $w \in (X \cup X^{-1})^*$ its normal form representation with respect to this free product, say $w = w_1 \cdots w_n$. By the normal form theorem for free products, $w \in K$ if and only if each $w_i \in \langle L \cap (X\backslash R)\rangle_G$ for each w_i from the second factor $F(X\backslash R)$. But $L \cap (X\backslash R)$ is a recursive subset of the generators of $F(X\backslash R)$ and hence we can effectively check these conditions.
In the second subcase, $R \nsubseteq L$. We distinguish two subsubcases. First assume that the stable letter t is omitted by L. Then, identifying all symbols $c \in X\backslash\{t\}$ with c_0, we see that under the isomorphism between G and the HNN extension of H, L is a subset of the set of generators of H. Hence the generalized word problem with respect to K in H is decidable. Now given any word $w \in (X \cup X^{-1})^*$, in order to algorithmically decide whether $w \in K$, first compute a representation of w in the HNN extension and effectively perform all stable letter reductions. By Britton's Lemma, if the reduced form still contains occurrences of t or t^{-1}, we can reject. Else, we can apply the algorithm that checks for elements of H whether they are in K and answer accordingly.
Finally, for the second subsubcase, assume that L contains the stable letter t (but omits some other generator from R). Let b be a generator from R that is not in L. W.l.o.g., we can assume that b is the generator with bounded subscript range from the construction (i.e., r starts with $b^{\pm 1}$). Now suppose we are given an element of G as a word $w \in (X \cup X^{-1})^*$. Note that since $t \in L$, $w \in K$ if and only if $w_1 := wt^{-\alpha} \in K$, where $\alpha := \sigma_t(w)$. Of course, the aim of this definition is to ensure that $\sigma_t(w_1) = 0$.
Now suppose for a moment that $w_1 \in K$. Then there also exists a representation of w_1 as a b-free word. Although we cannot assume that there is an algorithm that decides whether

there exists such a representation and, if so, outputs one (since such an algorithm would in particular decide the generalized word problem with respect to K), we can in theory fix one such representation if it exists and imagine all stable letter reductions carried out on the representation of this word in the HNN extension. Observe that the result is a t-free word since we can always perform reductions as long as there are occurrences of t and/or t^{-1} (because the t-balance is 0 and the letter with bounded subscript range does not appear). Hence in case $w_1 \in K$, it possesses in the HNN extension a t-free reduced form, which may not be algorithmically computable. However, by Britton's Lemma, this implies when carrying out the stable letter directions directly on the HNN representation of the word w_1, we must also end up with a t-free word!
This gives us the following algorithm for deciding whether $w_1 \in K$: Compute the HNN representation of w_1 and effectively perform all stable letter reductions. If the reduced form w_1^* that you obtain is not t-free, reject. Else, effectively check whether $w_1^* \in K^* := \langle c_i \ (c \in X \backslash \{t, b\}, i \in \mathbb{Z}) \rangle_H$, which is possible by the induction hypothesis. If the answer is "yes", accept, else reject. This completes the proof of the first case of the characteristic case distinction.
We are thus left with the second case, that is, all generators that occur in r have non-zero balance. Observe that since $\Psi : G \to C$ is a computable embedding and the HNN extension C, just as G in the first case, has decidable word problem, G also has decidable word problem. Hence the subcase $R \subseteq L$ works just as in the first case, and it remains to treat the subcase $R \nsubseteq L$. Since t is not distinguished by a special property from b (or any other generator occurring in r) any more, assume w.l.o.g. that $t \notin L$. But then the generators from L map under Ψ to powers of elements of the set $L' := (X \backslash \{t, b\}) \cup \{x\}$. By the Freiheitssatz, this is a free subset of C (because y occurs in r), whence the generalized word problem with respect to K in $K' := \langle L' \rangle_C$ is decidable. But as in the first case, the generalized word problem with respect to K' in C is also decidable; piecing together the two statements, we obtain that the generalized word problem with respect to K, as embedded by Ψ, in C is decidable. Since Ψ is computable, we obtain the desired result. □

5.2 Hyperbolic groups

This concept was introduced by Gromov in the 1980s. It falls into the vast area of geometric group theory, where groups are viewed as metric spaces to allow for the use of geometric methods in group theory. Of course, since this would very well serve as a topic for a thesis in its own right, we can only illustrate the main ideas. Also, recall that our point of interest in this thesis lies in the word problem, whence we will sketch just as much theory as is needed to understand the statement of the theorem of Gromov hinted at at the beginning of the chapter. As sources for this section up to the decidability of the word problem in hyperbolic groups, the Internet text (9) (where interested readers will find many futher references) as well as course notes from (1) were used. The author's attention for the stochastic considerations due to Gromov was raised by the Internet source (19).
First of all, how can we consider a group as a metric space?

Definition 5.2.1. *Let G be a group, $S \subseteq G$ a generating subset. For $g \in G$, define $|g|_S$ as the minimal $n \in \mathbb{N}$ such that there exists a representation of g as a product of n elements of S or inverses of elements of S. The* ***word metric with respect to S on G*** *is defined as the function $d_S : G^2 \to \mathbb{N}, (g_1, g_2) \mapsto d(g_1, g_2) := |g_1^{-1} g_2|_S$.*

It is easy to see that d_S really is a metric on the set G. This metric takes values in the

discrete subset $\mathbb{N}$ of $\mathbb{R}$, but we can view it as a restriction of a "continuous" metric on a bigger set as follows: Define the **Cayley graph** of G with respect to S, denoted $\Gamma(G,S)$, as the (undirected) graph with G as the set of vertices such that for all $g_1, g_2 \in G$, $\Gamma(G,S)$ contains precisely one edge between g_1 and g_2 if and only if $g_1^{-1}g_2 \in S$ or $g_2^{-1}g_1 \in S$, and otherwise no edge. Then it is not difficult to see that the distance in the path-metric between two elements g_1 and g_2 of G, viewed as vertices of $\Gamma(G,S)$, is exactly $d_S(g_1,g_2)$. Now we can associate with the abstract graph $\Gamma(G,S)$ a path-connected metric space which corresponds to $\Gamma(G,S)$ "as a whole", by taking for each edge e of $\Gamma(G,S)$ a copy of the compact unit interval $[0,1]$ and gluing the copies at endpoints appropriately, defining a metric on the resulting set in the most obvious way to fit the intention.
Of course, the metric space (G,d_S) and the metric space just constructed (which we will, by abuse of notation, also denote by $\Gamma(G,S)$) cannot be isometric (because the set of values taken by the according metrics are different), but they look "similar from a distance". To make this more precise, we define:

Definition 5.2.2. *Let (X,d_X) and (Y,d_Y) be metric spaces, $K \geqslant 1$ and $L \geqslant 0$ real constants.*
*(1) A (K,L)-**quasi-isometric embedding of** X **into** Y is a map $f: X \to Y$ such that for all $x_1, x_2 \in X$: $K^{-1} \cdot d_Y(f(x_1),f(x_2)) - L \leqslant d_X(x_1,x_2) \leqslant K \cdot d_Y(f(x_1),f(x_2)) + L$.*
*(2) A subset $D \subseteq Y$ is called L-**dense in** Y if and only if for each $y \in Y$, there exists some $d \in D$ such that $d_Y(d,y) \leqslant L$.*
*(3) A (K,L)-quasi-isometric embedding of X into Y whose image is L-dense in Y is called a (K,L)-**quasi-isometry between** X **and** Y. X and Y are called quasi-isometric if and only if there exists a quasi-isometry between X and Y.*

For example, the map $G \to \Gamma(G,S)$, mapping $g \in G$ to the point of $\Gamma(G,S)$ corresponding to the vertex g of the abstract graph $\Gamma(G,S)$, is a $(1,0)$-quasi-isometric embedding (that is, an isometric embedding in the usual sense) and even a $(1,\frac{1}{2})$-quasi-isometry. Hence up to quasi-isometry, the metric spaces G and $\Gamma(G,S)$ are the same. Furthermore, we can show:

Proposition 5.2.3. *Let G be a group, S_1, S_2 two finite generating subsets of G. Then (G,d_{S_1}) and (G,d_{S_2}) are quasi-isometric.*

Proof. Of course, the map of which we want to show that it is a quasi-isometry is the identity map on G. We will be able to take $L = 0$. For the multiplicative constant K, set $m_1 := \max\{|s_1|_{S_2} \mid s_1 \in S_1\}$, $m_2 := \max\{|s_2|_{S_1} \mid s_2 \in S_2\}$, and $K := \max\{m_1, m_2\}$. Then it is clear that for all $g \in G$, we have $|g|_{S_1} \leqslant K \cdot |g|_{S_2}$ and $|g|_{S_2} \leqslant K \cdot |g|_{S_1}$, whence $K^{-1} \cdot |g|_{S_2} \leqslant |g|_{S_1} \leqslant K \cdot |g|_{S_2}$. Looking at the definition of "word metric", we see that we are done. □

Of course, the statement of Proposition 5.2.3 is not true when at least one of S_1 and S_2 is allowed to be infinite. For example, setting $S_1 := G$, d_{S_1} turns into the discrete metric on G. In particular, (G,d_{S_1}) is bounded. However, any finitely generated infinite group G, when viewed as a metric space with respect to some finite generating set S_2, is unbounded, and it is clear that a bounded and an unbounded metric space cannot be quasi-isometric.
For the definition of hyperbolic groups, we first define what a hyperbolic metric space is; this concept itself builds up on the following definition:

Definition 5.2.4. *Let (X,d) be a metric space.*
*(1) A **geodesic in** X is a quasi-isometric embedding of a real compact interval into X.*

(2) For $a, b \in \mathbb{R}$ with $a \geqslant b$ and $\gamma : [a, b] \to X$ a geodesic in X, $\gamma(a)$ and $\gamma(b)$ are called its ***endpoints****. We also say that γ is* ***from*** *$\gamma(a)$* ***to*** *$\gamma(b)$.*
(3) (X, d) is called ***geodesic*** *if and only if for all $x, y \in X$, there is a geodesic from x to y in X.*

Note that for (X, d) to be geodesic, geodesics from x to y need not be unique. For example, every connected graph, viewed as a path-connected metric space as described above, is geodesic, with geodesics between vertices corresponding to shortest paths connecting them, and shortest paths are in general not unique (consider a cycle of even length). Trees are examples of graphs where the corresponding path-connected metric space has a unique geodesic between every two points.
One can also consider the standard examples of geometry such as the Euclidean plane $\mathbb{E}^2$ or the hyperbolic plane $\mathbb{H}^2$. Both are geodesic metric spaces, but there are of course also crucial differences. One of them is that in $\mathbb{H}^2$, all triangles formed by choosing three different points and for each of the three pairs of points a geodesic between them (so-called **geodesic triangles**) are "thin", in the sense that for all geodesic triangles in $\mathbb{H}^2$, each of the three sides of the triangle is contained in the closed 1-neighborhood of the union of the other two; now this directly generalizes as:

Definition 5.2.5. *Let (X, d) be a geodesic metric space.*
*(1) For $\delta \geqslant 0$, a geodesic triangle in X is called δ-****thin*** *if and only if each of the three sides of the triangle is contained in the closed δ-neighborhood of the union of the other two.*
*(2) (X, d) is called δ-****hyperbolic*** *for some constant $\delta \geqslant 0$ if and only if all geodesic triangles in X are δ-thin, and* ***hyperbolic*** *if and only if it is δ-hyperbolic for some $\delta \geqslant 0$.*

Clearly, $\mathbb{E}^2$ is not hyperbolic, whereas trees are 0-hyperbolic. Also, all bounded metric spaces trivially are hyperbolic.
The following lemma, the proof of which is quite technical and can be found in (19), states that hyperbolicity is a "coarse" property of metric spaces, i.e., one that is preserved not just under isometry, but even under quasi-isometry:

Lemma 5.2.6. *Let (X, d_X) and (Y, d_Y) be quasi-isometric geodesic metric spaces. Then X is hyperbolic if and only if Y is hyperbolic.* □

We now define:

Definition 5.2.7. *A finitely generated group G is called* ***hyperbolic*** *if and only if for some (and hence, by Lemma 5.2.6 and Proposition 5.2.3, any) finite generating subset S of G, $\Gamma(G, S)$ is a hyperbolic metric space.*

That trees are hyperbolic translates as "free groups are hyperbolic" and that bounded metric spaces are as "finite groups are hyperbolic".
Of course, an at first glance "unconventional" definition like this (why just study this property?) draws its life from the consequences provable. It is clear by the character of the definition that proofs working with it will use geometric methods. As a first example, we show:

Theorem 5.2.8. *All hyperbolic groups are finitely presented.*

Proof. Let G be a hyperbolic group, and fix a finite generating subset S of G. Note that we can obtain a presentation for G by taking for each element $g \in G$ a letter x_g and adding the "multiplication table relations" $x_{gh} = x_g \cdot x_h$. Since we are interested in finding a finite presentation for G, we cannot take this presentation directly, but try to approximate it by finite presentations. For $n \in \mathbb{N}\backslash\{0\}$, we set $X_n := \{x_g \mid g \in G, |g|_n \leqslant n\}$ and $R_n := \{x_{gh}x_g^{-1}x_h^{-1} \mid g, h \in X_n, gh \in X_n\}$. Now let $G_n := \langle X_n \mid R_n \rangle$, and note that we have canonical homomorphisms $\varphi_n : G_n \to G_{n+1}$, giving us a direct system of finitely presented groups with underlying directed poset $(\mathbb{N}\backslash\{0\}, \leqslant)$ whose direct limit clearly is G.
Also, observe that all the canonical homomorphisms even are surjective, since the generators from $X_{n+1}\backslash X_n$ are made superfluous in G_{n+1} by corresponding multiplication table relations which express them (in various ways) as products of two of the "old" generators.
So far, this is a general construction that works for every finitely generated group G. We will now use the hyperbolicity of G to show that for sufficiently large N, φ_N actually is an isomorphism, so that $G \cong G_N$ for any of these N, and we will be done. "Sufficiently large" more precisely means $N \geqslant 2(\lfloor\delta\rfloor + 1)$, where $\delta \geqslant 0$ is such that G is δ-hyperbolic.
So fix such an N. We first remove the generators lying in $X_{N+1}\backslash X_N$ from the presentation $\langle X_{N+1} \mid R_{N+1} \rangle$ by a finite sequence of Tietze transformations, as follows: Choose (you do not need AC for this, because there are only finitely many choices to make) for each $g \in G$ with $x_g \in X_{N+1}\backslash X_N$, a **canonical splitting** $g = g_1g_2$, where $x_{g_1}, x_{g_2} \in X_N$, $|g|_S = |g_1|_S + |g_2|_S$ and $|g_1|_S, |g_2|_S > \delta$ (note that there is an according relation giving us the splitting). Now delete the generator x_g and replace any of its occurrences in any of the relations from R_{N+1} by $x_{g_1}x_{g_2}$. Continue until all "new" generators are deleted.
We now have a presentation of G_{N+1} in terms of only the generators from X_N, and the corresponding set of relations is a superset of R_N. Now if we can show that any of the other relations (which were obtained by substituting canonical splittings into relations from $R_{N+1}\backslash R_N$) already follow from the relations in R_N, we are done. So consider any relation $x_g = x_hx_k$ from $R_{N+1}\backslash R_N$. At least one of g, h, k is not in the closed N-ball of 1_G. Note that the relations from R_N allow us to identify the letter standing for 1_G (since 1_G is the only idempotent in G) and hence also, for each $x \in X_N$, which letter stands for the inverse element. Hence the relation $x_g = x_hx_k$ is equivalent under R_N to $x_h = x_gx_{k^{-1}}$ in case $x_k \in X_N$ and to $x_k = x_{h^{-1}}x_g$ in case $x_h \in X_N$. We distinguish two cases:
First, assume that precisely one of g, h, k is not in the closed N-ball of 1_G. By the observations from the last paragraph, we can w.l.o.g. in this case even assume that $x_g \notin X_N$, and $x_h, x_k \in X_N$. Note that the canonical splitting $g = g_1g_2$ gives us a geodesic joining 1_G and g which is split into two parts by the point P. Consider the triangle Δ in $\Gamma(G, S)$ with vertices 1_G, h and g such that the side joining 1_G and g is this geodesic, and 1_G and h are joined by any path of length $|h|_S$, and h and g by any path of length $|k|_S$. Since G is δ-hyperbolic, there is a point Q on one of the two other sides of Δ which is at distance at most δ from P. Then any geodesic joining P and Q together with any geodesic joining Q with the vertex of Δ opposite to the edge of Q divide Δ into three smaller triangles. It is not difficult to see that any of the sides of these three triangles have length at most N, so that they correspond to three relations from R_N which together imply $x_g = x_hx_k$.
In the second case, at least two of g, h, k have distance more than N to 1_G; w.l.o.g., we assume that these are k and g. We consider the same triangle Δ as in the first case, with the same splitting point P and point Q. Note that now, Q cannot coincide with the point h because otherwise, we could travel "fast" (in distance less than $N + 1$) from h to g, contradicting the fact that the path corresponding to k also joins these two points, has length $N + 1$ and is

geodesic. Now it may be that Q lies on a side of length $N+1$, in which case it gives us a (possibly non-canonical) proper splitting of h or k. In any case, the relation corresponding to the splitting induced by Q is one that we are allowed to assume (possibly needing to invoke the first case). Next, divide Δ into three smaller triangles as before. Now one of the sides of these may have length $N+1$, but all other certainly have length at most N, so that the three relations given by the triangles are ones that we may assume, and we are done. □

The converse of Theorem 5.2.8 is easily seen to be false; for example, the finitely presented group $\mathbb{Z}^2$ is not hyperbolic. However, Gromov's Theorem on hyperbolicity in randomly chosen finitely presented groups states that it is "almost" correct.
We now turn to the word problem in hyperbolic groups. Again, the proof builds up on geometric ideas. In elementary differential geometry courses, one learns that (assuming for simplicity that we live on a totally flat surface) as far as area is concerned, when a farmer has to decide for the best form for a piece of pasture for their cattle to be bounded by a fence of a given length, the solution is a circle:

Theorem 5.2.9. *(Isoperimetric Inequality for the Euclidean plane)*
Let C be a smooth simply closed curve in $\mathbb{E}^2$. Denote by $L(C)$ the length of C and by $A(C)$ the area of the domain bounded by C. Then $A(C) \leqslant \frac{L(C)^2}{4\pi}$, and equality is assumed precisely when C is a circle line. □

There also is a corresponding theorem in the hyperbolic case:

Theorem 5.2.10. *(Isoperimetric Inequality for the Hyperbolic plane)*
Let C be a smooth simply closed curve in $\mathbb{H}^2$, $L(C)$ and $A(C)$ as in Theorem 5.2.9. Then $A(C) \leqslant L(C)$, and equality is assumed precisely when C is a (hyperbolic) circle line. □

So in the hyperbolic case, the upper bound for $A(C)$ is linear in $L(C)$. This idea carries over to groups as well. Let G be a finitely presented group, and fix a finite presentation $\langle X \mid R\rangle$ for G. Recall that the set of all relators that hold in G is precisely the normal subgroup of $F(X)$ generated by R. Hence every such relator w is a product of conjugates of elements of R and their inverses. The minimal number of such factors required to represent w is called the **area of** w, denoted $\mathrm{Area}(w)$. There is a reason for this nomenclature; first, we define:

Definition 5.2.11. *Let $\mathcal{P} = \langle X \mid R\rangle$ be a finite group presentation. A **van Kampen diagram over** $\mathcal{P}$ consists of:*
(1) A simply connected, finite 2-dimensional CW complex M embedded into the plane.
(2) An orientation of each 1-cell of M.
(3) A labeling function that assigns to each 1-cell of M a label in X such that for each 2-cell, the label read from the boundary of the cell from some starting point in some direction is an element of R, where each 1-cell of the boundary labeled with x contributes x to the boundary label when read in orientation direction and x^{-1} otherwise.

Those unfamiliar with CW complexes may think of van Kampen diagrams as embeddings of finite, planar, directed, connected, X-labeled graphs into the plane $\mathbb{R}^2$ without crossings of edges and such that edges which do not lie on any (chordless) cycle in the graph are not drawn inside one of the domains surrounded by a chordless cycle (if such a drawing is not possible, then the graph does not represent a CW complex). The 0-cells of the complex correspond to the vertices of the graph, the 1-cells to the edges and the 2-cells to the domains

bounded by chordless cycles of the graph. Note that the complement of M in the plane also has a boundary which corresponds to some closed path in this graph. Its label, which is only defined up to cyclic permutation and formal inversion, is called the **boundary label** of M. The following lemma, which we give without proof, is the raison d'être for van Kampen diagrams:

Lemma 5.2.12. *Let $\mathcal{P} = \langle X \mid R \rangle$ be a finite presentation of the group G. Then for all $w \in F(X)$, there exists a van Kampen diagram over $\mathcal{P}$ whose boundary label, viewed as an element of $F(X)$, equals w if and only if $w = 1$ in G, in which case $\mathrm{Area}(w)$ is the minimum number of 2-cells in such a van Kampen diagram.* □

Fix a finite group presentation $\mathcal{P} = \langle X \mid R \rangle$. Then we can assign to this presentation a function $f_{\mathcal{P}} : \mathbb{N} \to \mathbb{N}$ via $n \mapsto \max\{\mathrm{Area}(w) \mid w \in F(X), w = 1 \text{ in } G, |w| \leqslant n\}$. $f_{\mathcal{P}}$ is called the **Dehn function** (or **isoperimetric function**) for $\mathcal{P}$. Note that $f_{\mathcal{P}}$ really depends on $\mathcal{P}$ and not just on the abstract group which is presented by $\mathcal{P}$ (in contrast to, for instance, the decidability of the word problem). However, the Dehn functions for different finite presentations of one group are "roughly the same":

Lemma 5.2.13. *Let $\mathcal{P}$ and $\mathcal{Q}$ be finite presentations of the same group G. Then there are $A, B, C \in \mathbb{N}$ such that for all $n \in \mathbb{N}$, $f_{\mathcal{P}}(n) \leqslant A \cdot f_{\mathcal{Q}}(Bn) + Cn$.*

Proof. Let w be a reduced word over the generators from $\mathcal{P}$ and their formal inverses of length $l \leqslant n$ which is the identity in G. We need to appropriately bound $\mathrm{Area}_{\mathcal{P}}(w)$. Fix once and for all a representation of each generator from $\mathcal{P}$ as a word over the generators from $\mathcal{Q}$ and their formal inverses (denoting by B the maximum length of such a representation) and vice versa. Denote by A the maximum area of one of the words obtained by replacing in a defining relator from $\mathcal{Q}$ each occurrence of each generator by the according fixed representation over $\mathcal{P}$.
We can associate to each generator x from $\mathcal{P}$ a "blown-up" version w_x of x obtained by first replacing x by the corresponding word over the generators from $\mathcal{Q}$ and then replacing in this word each occurrence of any generator x' from $\mathcal{Q}$ by the corresponding word over the generators from $\mathcal{P}$. Since w_x and x denote the same element of G, $\mathrm{Area}(w_x x^{-1})$ is well-defined; let C be the maximum among the values $\mathrm{Area}(w_x x^{-1})$, where x ranges over the generators from $\mathcal{P}$.
Now by substituting the representations of the generators from $\mathcal{P}$ over $\mathcal{Q}$ into w and performing reductions if necessary, we obtain a representation $\tilde{w}$ of the element of G represented by w in terms of the generators from $\mathcal{Q}$ which has length at most $B \cdot l \leqslant B \cdot n$. Fix a representation of $\tilde{w}$ as a product of $\mathrm{Area}_{\mathcal{Q}}(\tilde{w})$ conjugates of the defining relators from $\mathcal{Q}$; these are at most $f_{\mathcal{Q}}(Bn)$ factors. Replacing the occurrences of the generators from $\mathcal{Q}$ in this word by their representations over $\mathcal{P}$, we get a representation (in the free group over the generators of $\mathcal{P}$) of the blown-up version of w (i.e., if $w = x_1^{\epsilon_1} \cdots x_n^{\epsilon_n}$, of the word $w_{x_1}^{\epsilon_1} \cdots w_{x_n}^{\epsilon_n}$) as some product of at most $f_{\mathcal{Q}}(Bn)$ conjugates of words which represent 1_G. By choice of A, we can write each of these words as a product of at most A conjugates of the defining relations from $\mathcal{P}$. Hence, we have found a representation of the blown-up version of w as a product of at most $A \cdot f_{\mathcal{Q}}(Bn)$ conjugates of defining relators. In order to pass from this representation to an according representation of w itself, by choice of C, we can add a certain number, bounded by Cn, of conjugates of defining relators from $\mathcal{P}$ (observe that, as is easy to see, $\mathrm{Area}(w_{x_1}^{\epsilon_1} \cdots w_{x_n}^{\epsilon_n} (x_1^{\epsilon_1} \cdots x_n^{\epsilon_n})^{-1}) \leqslant \mathrm{Area}(w_{x_1}^{\epsilon_1} x_1^{-\epsilon_1}) + \cdots + \mathrm{Area}(w_{x_n}^{\epsilon_n} x_n^{-\epsilon_n})$). □

In particular, if one of the Dehn functions corresponding to the finite presentations of some group G can be bounded above by a linear function, then this is true for all of them. This motivates:

Definition 5.2.14. *Let G be a finitely presented group. We say that G* ***has a linear isoperimetric inequality*** *if and only if for some (hence, any) finite presentation of G, the corresponding Dehn function can be bounded above by a linear function.*

Using geometric methods, one can show:

Theorem 5.2.15. *Let G be a finitely presented group. The following are equivalent:*
(1) G is hyperbolic.
(2) G has a linear isoperimetric inequality. □

Corollary 5.2.16. *Any hyperbolic group has decidable word problem.*

Proof Sketch. Let G be a hyperbolic group, and by Theorem 5.2.8, fix a finite presentation $\mathcal{P} = \langle X \mid R \rangle$ of G. By Theorem 5.2.15, fix a linear function $f : \mathbb{N} \to \mathbb{N}$ such that for all $n \in \mathbb{N}$, we have $f_{\mathcal{P}}(n) \leqslant f(n)$. An algorithm deciding the word problem for $\mathcal{P}$ works as follows: Given some input $w \in (X \cup X^{-1})^*$, it makes reductions in w until it has found the reduced form $\overline{w}$. Then it computes $f(n)$, where n is the length of the reduced word $\overline{w}$. To make the proof complete, one would now have to show that there are only finitely many van Kampen diagrams over $\mathcal{P}$ of a given area and boundary length (if $w = 1$ in G, there is a van Kampen diagram over $\mathcal{P}$ whose boundary label in $(X \cup X^{-1})^*$ is $\overline{w}$; one can obtain such a diagram from any whose boundary label is a word equal to $\overline{w}$ in $F(X)$ without changing the area by appropriate manipulations) and that there is an effective way to check the (hence finitely many) relevant cases. □

We close this thesis with some stochastic considerations due to Gromov himself; see also (6) (the beginning of section 0.2(A) on p.78), (7) and, for a survey article, (14).
For the rest of this text, we fix the number of generators $m \in \mathbb{N}$. The historically first model for studying random groups was the following:

Definition 5.2.17. *(The few-relator model)*
Fix $k, l \in \mathbb{N}$, and let $\mathcal{R}_{k,l}$ be the set of all group presentations of the form $\langle x_1, \ldots, x_m \mid r_1, \ldots, r_k \rangle$, where $r_1, \ldots, r_k$ are reduced words over the x_i and their inverses of length at most l. Observe that $\mathcal{R}_{k,l}$ is finite. We say that some property of (finite) group presentations ***occurs with overwhelming probability in presentations with k relators*** *if and only if the ratio of members of $\mathcal{R}_{k,l}$ that have this property tends to 1 as $l \to \infty$.*

The key to showing that a finitely presented group is hyperbolic with overwhelming probability in the sense of Definition 5.2.17 lies in small cancellation theory, which already existed before the introduction of hyperbolic groups by Gromov (see (10)):

Definition 5.2.18. *Let R be a subset of $F(x_1, \ldots, x_m)$, its elements represented as reduced words.*
(1) R is called ***symmetrized*** *if and only if the elements of R are cyclically reduced, and for each $r \in R$, all cyclically reduced conjugates of r and of r^{-1} also belong to R.*
(2) If R is symmetrized, and $r_1, r_2 \in R$ are distinct and of the form $r_1 = bc_1, r_2 = bc_2$ in $(\{x_1, \ldots, x_m, x_1^{-1}, \ldots, x_m^{-1}\})^*$, where b, c_1, c_2 are words over the x_i and their inverses, then b is called a ***piece relative to*** *R.*

(3) Let $\lambda > 0$, $R \subseteq F(x_1, \ldots, x_m)$ symmetrized. We say that the presentation $\langle x_1, \ldots, x_m \mid R \rangle$ is $C'(\lambda)$ if and only if for all $r \in R$ and all pieces b relative to R such that $r = bc$ for some word c in the free monoid over the x_i and their inverses, then $|b| < \lambda |r|$.
(4) For $\lambda > 0$, a finitely presented group is called $C'(\lambda)$ if and only if it has a finite presentation which is $C'(\lambda)$.

Obviously, pieces are subwords that can be cancelled by appropriately multiplying noninverse elements of R; the small cancellation hypothesis $C'(\lambda)$ for "small" λ states that pieces are not long, so we can never cancel much as long as we keep concatenating elements of R. It is not difficult to see:

Proposition 5.2.19. *For $R \subseteq F(x_1, \ldots, x_m)$, let R^{symm} denote the **symmetrization of** R, i.e., the subset of $F(x_1, \ldots, x_m)$ obtained by replacing each element of R by one of its cyclically reduced forms and then closing under inverses and cyclically reduced conjugates. For any presentation $\mathcal{P} = \langle x_1, \ldots, x_m \mid r_1, \ldots, r_k \rangle$, we call the presentation obtained by replacing $R = \{r_1, \ldots, r_k\}$ by its symmetrization the **symmetrization of** $\mathcal{P}$ (note that this is another presentation of the same group). Then for every $\lambda > 0$ and every $k \in \mathbb{N}$, with overwhelming probability the symmetrization of a finite presentation with k relators is $C'(\lambda)$.* □

However, one can also show:

Theorem 5.2.20. *Let G be a $C'(\frac{1}{6})$ finitely presented group. Then G is hyperbolic.* □

As a consequence, we now obtain:

Corollary 5.2.21. *For any $k \in \mathbb{N}$, the following are valid:*
(1) Presentations with k relators represent hyperbolic groups with overwhelming probability.
(2) Presentations with k relators have decidable word problem with overwhelming probability.

Proof. (1) follows immediately from Proposition 5.2.19 and Theorem 5.2.20, and (2) from (1) by Corollary 5.2.16. □

Note that the finitely presented groups with undecidable word problem which we have constructed in Chapter 4 have enormously big finite presentations. In view of this and Corollary 5.2.21, one may believe that all finite presentations with undecidable word problem must be "monsters" which one may find very difficult to write down explicitly. However, this is not the case; in 1986, Collins gave an example of a finite group presentation with 10 generators and 27 relators which has undecidable word problem and can be written down in a few lines, see (4) and note that there is a misprint in the group presentation there; the correct version of the presentation can be found in (15).
We conclude by noting that there are other models for random finitely presented groups, like the density model dependent on a parameter d with $0 \leqslant d \leqslant 1$ also introduced by Gromov; much more on these can be found in (14).

References

[1] G.N. Arzhantseva and D. Osajda. Geometric and asymptotic group theory. Notes of the lecture held at Vienna University in the fall of 2013.

[2] K. Auinger. Gruppentheorie. Notes of the lecture held at Vienna University in the spring of 2013.

[3] W.W. Boone. The word problem. *Proc. Nat. Acad. Sci. U.S.A.*, **44**:265–269, 1958.

[4] D.J. Collins. A simple presentation of a group with undecidable word problem. *Illinois Journal of Mathematics*, **30**(2):230–234, 1986.

[5] M. Dehn. Über unendliche diskontinuierliche Gruppen. *Math. Ann.*, **71**:116–144, 1912.

[6] M.L. Gromov. Hyperbolic groups. *Essays in group theory*, pages 75–265, 1987.

[7] M.L. Gromov. Random Walk in Random Groups. *Geom. funct. anal.*, **13**:73–146, 2003.

[8] H. Hermes. *Aufzählbarkeit; Entscheidbarkeit; Berechenbarkeit.* Springer, Berlin, 2. edition, 1971.

[9] J. Howie. Hyperbolic Groups. Lecture Notes. `http://www.macs.hw.ac.uk/~jim/samos.pdf` (24.4.2014).

[10] R.C. Lyndon and P.E. Schupp. *Combinatorial Group Theory.* Springer, Berlin Heidelberg New York, 1977.

[11] Y.V. Matiyasevich. *Hilbert's Tenth Problem.* MIT Press, Cambridge London, 1993.

[12] M. Müller. Introduction to theoretical computer science. Script of the lecture held at Vienna University (KGRC) in the fall of 2013. An online version is available under `http://www.logic.univie.ac.at/~muellem3/v3.pdf`.

[13] P.S. Novikov. On the algorithmic unsolvability of the word problem in group theory. *Trudy Mat. Inst. Steklov*, **44**:1–143, 1955.

[14] Y. Ollivier. A January 2005 Invitation to Random Groups. `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.417.7016&rep=rep1&type=pdf` (1.5.2014), 2005.

[15] J. Pedersen. A Catalogue of Algebraic Systems: Groups. `http://shell.cas.usf.edu/~wclark/algctlg/groups.html` (1.5.2014).

[16] C. Tornau. Grundstudium.info: Registermaschine ↔ Turingmaschine. `http://www.grundstudium.info/theorie/node9.php` (13.6.2014).

[17] A. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, **42**(2):230–265, 1937.

[18] M.K. Valiev. One theorem of G. Higman. *Algebra i Logika*, **9**:9–22, 1968.

[19] H. Wilton. 392C Geometric Group Theory. `http://392c.wordpress.com/` (24.4.2014).

Zusammenfassung

Diese Masterarbeit behandelt das Wortproblem für Gruppen; in drei einleitenden Kapiteln werden der Reihe nach die rekursionstheoretischen (Turing-Maschinen), zahlentheoretischen (Satz von Matiyasevich über Diophantizität und rekursive Aufzählbarkeit) sowie gruppentheoretischen (kombinatorische Gruppentheorie) Grundlagen für die Konstruktion endlicher Gruppenpräsentationen mit unentscheidbarem Wortproblem entwickelt. Im vierten Kapitel wird mit Hilfe dieser Grundlagen nicht nur diese Konstruktion besprochen, sondern auch der Einbettungssatz von Higman bewiesen. Das zur Abrundung dienende fünfte Kapitel bespricht zwei Klassen von Gruppen mit entscheidbarem Wortproblem, nämlich Ein-Relator-Gruppen und hyperbolische Gruppen.

Abstract

This Master's thesis deals with the word problem for groups; in three preliminary chapters, we successively develop the recursion-theoretic (Turing machines), number-theoretic (Matiyasevich's Theorem on Diophantineness and recursive enumerability) and group-theoretic (combinatorial group theory) foundations for the construction of finite group presentations with undecidable word problem. In the fourth chapter we discuss not only this construction by means of these foundations, but also prove the Higman Embedding Theorem. The fifth chapter, which serves to round off the thesis, discusses two classes of groups with decidable word problem, namely one-relator groups and hyperbolic groups.

Lebenslauf

Persönliche Daten

Name	Alexander Bors
E-Mail	alexander.bors@gmx.at
Geburtsdaten	24. März 1991 in Bad Ischl
Staatsbürgerschaft	Österreichisch

Schulische Ausbildung

09/1997 – 07/2001	**Volksschule 1 Bad Aussee**.
09/2001 – 07/2005	**Hauptschule 2 Bad Aussee**.
09/2005 – 07/2009	**Erzherzog-Johann-BORG Bad Aussee**, *Matura mit ausgezeichnetem Erfolg*.

Universitäre Ausbildung

10/2009 – 07/2012	**Bachelor-Studium der Mathematik**, *Paris-Lodron-Universität Salzburg*, Abgeschlossen (ausgezeichneter Erfolg).
10/2012 – heute	**Master-Studium der Mathematik mit Schwerpunkt „Algebra, Zahlentheorie und Diskrete Mathematik“**, *Universität Wien*.
geplant ab 10/2014	**Doktorats-Studium der Mathematik an der Paris-Lodron-Universität Salzburg unter Betreuung durch Ao. Univ.-Prof. Dr. Peter Hellekalek**, *Mitarbeit im SFB „Quasi-Monte Carlo Methods: Theory and Applications“*.

Auszeichnungen

09/2009	1. Schülerpreis der ÖMG für die FBA „Die Funktion, die ins Komplexe ging. Die Diskussion holomorpher komplexer Funktionen“

Abschlussarbeiten

A. Bors. **Von Körpern, Gruppen und Gleichungen. Eine Einführung in die Galois-Theorie.** Bachelorarbeit, Paris-Lodron-Universität Salzburg, 2012.

A. Bors. **On the Word Problem for Groups.** Masterarbeit, Universität Wien, 2014.